# UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

**Escuela de Matemáticas y Ciencias Computacionales**

## Deep Reinforcement Learning for Efficient Nucleus Cell Location in Digital Pap Smears

Trabajo de integración curricular presentado como requisito para la obtención del título de ingeniero en tecnologías de la información


**Autor:**

Macancela Bojorque Carlos Julio

**Tutor:**

Ph.D. Morocho Cayamcela Manuel Eugenio


Urcuquí, Junio 2023

# Autoría

Yo, **Macancela Bojorque Carlos Julio**, con cédula de identidad 0105619498, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Junio del 2023.


Carlos Julio Macancela Bojorque

CI:0105619498

# Autorización de publicación

Yo, **Macancela Bojorque Carlos Julio**, con cédula de identidad 0105619498, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Junio del 2023.

Firmado electrónicamente por:
CARLOS JULIO
MACANCELA BOJORQUE

Carlos Julio Macancela Bojorque

CI:0105619498

# Dedication

To my lovely family, Carlos, Ely and Abraham, who always believe in me and support me wherever and whenever. To my dear little dog Paquito who has seen me grow up and will now see me graduate from the college. Last but not least, to my beautiful Lucy, who is my new little dog sister.

Carlos Julio Macancela Bojorque

# Acknowledgment

Thanks to my parents Carlos and Ely for your unwavering faith in me and unconditional support, you have been my pillars of strength, motivating me to achieve my goals and dreams. Thanks to my brother Abraham for having the courage to open paths and always being my competition. Overall, I extend my gratitude to each and every member of my family who believe in me.

I want to express my immensely grateful to my advisors Ph.D. Oscar Chang and Ph.D. Eugenio Morocho who influence a lot in my academic development, as well as for the opportunity to work with them on this thesis. I want to mentioned that despite the unfortunate disassociation of my professor Oscar Chang from the university, I extend my heartfelt wishes for his future prosperity and a fulfilling life.

Besides, in reflecting upon the last two semesters of my college career, which remind me of the oft-quoted phrase "University is the best stage of your life." I am now certain that this statement holds true. I was fortunate enough to know incredible people, such as "Los Pibes de la Pat. 28" composed of Mister Vegancio, Christian Caile, Jorry Solano, Jorgito, Luis, Francis and Eder with whom I have spent great moments, whether in study, sports or especially recreational. In general, I am grateful for everyone I have had the pleasure of meeting and interacting with during my time in university.

Finally, I must highlight the significance of meeting my beautiful girlfriend Dani in this final stage of university. By some twist of fate, we crossed paths and now I am incredibly fortunate to have her in my life. She provides me with unconditional support, love, charm, and above all, her intelligence is truly awe-inspiring. I cannot express enough how much I love her and her company.

Carlos Julio Macancela Bojorque

# Resumen

En agosto de 2020, la Asamblea Mundial de la Salud (con siglas en inglés WHA) definió tres objetivos mundiales principales para eliminar el cáncer de cuello uterino. Con suerte, será posible eliminar el cáncer de cuello uterino para 2030 siguiendo estos objetivos. Uno de esos objetivos es "el 70% de cobertura de cribado". Este objetivo requería profesionales experimentados para completar el análisis de las imágenes digitales de Papanicolaou. Los patólogos llevan a cabo el análisis de las imágenes, lo que lleva alrededor de 30 minutos por prueba. La falta de patólogos especialistas retrasa las metas propuestas por la WHA. Esta tesis se enfoca en utilizar un agente de aprendizaje por refuerzo profundo que aprende por sí mismo, mediante recompensas, penalizaciones y experiencias pasadas, para avanzar hacia el núcleo celular en imágenes digitales, siguiendo un camino óptimo. En principio, el resultado de la tesis será crear un agente cuya entrada sean píxeles en bruto y su salida sean coordenadas del núcleo. Para futuros trabajos, esta información será utilizada por otros agentes especializados o redes neuronales para detectar células desviadas o con anomalías. La idea final es construir una máquina de análisis de Papanicolaou automática de alta eficiencia.

**Palabras Clave**: Aprendizaje de refuerzo profundo, redes neuronales, prueba de Papanicolaou, cancer cervical.

# Abstract

In August 2020, World Health Assembly (WHA) defined three main global targets for eliminating cervical cancer. Hopefully, they may eliminate cervical cancer by 2030 by following three targets. One of those targets is "70% coverage of screening". This target required experienced professionals to complete the analysis of Papanicolaou digital images. Pathologists carry out the analysis of the image which takes around 30 minutes. The lack of pathologists retard the goals of the target proposed by WHA. This thesis focuses on using a deep reinforcement learning agent that learns by itself, by rewards, penalties and pass experiences, to move toward cell nucleus in digital images, by following an optimal path. In principle this will create an agent whose input are raw pixels and its output is nucleus coordinates. This information will be used for other specialised future agents to detect deviated cells. The final idea is to construct high efficiency automatic Papanicolaou analyzing machine.

**Keywords**: Deep reinforcement learning, neuronal network, Pap smear, cervix cancer.

# Contents

**Dedication** **v**

**Acknowledgment** **vii**

**Resumen** **ix**

**Abstract** **xi**

**Contents** **xiii**

**List of Tables** **xvii**

**List of Figures** **xix**

**1 Introduction** **1**

1.1 Background . . . . . 1

1.2 Problem Statement . . . . . 2

1.3 Objectives . . . . . 3

1.3.1 General Objective . . . . . 3

1.3.2 Specific Objectives . . . . . 4

**2 Theoretical Framework** **5**

2.1 Reinforcement Learning . . . . . 5

2.1.1 Environment . . . . . 5

2.1.2 Agent . . . . . 6

2.1.3 Policy . . . . . 7

2.1.4 Reward Signal . . . . . 7

2.1.5 Value Function . . . . . 8

2.1.6 Episode . . . . . 8
2.1.7 Exploitation and Exploration . . . . . 9
2.1.8 $Q$-learning . . . . . 9
2.2 $Q$-function Approximation . . . . . 11
2.2.1 Feedforward Neuronal Networks . . . . . 11
2.2.2 Convolutional Neuronal Network . . . . . 13
2.3 Data Augmentation . . . . . 17
2.4 Proximal Policy Optimization Clip . . . . . 17
2.4.1 Policy Gradient . . . . . 17
2.4.2 Clipping . . . . . 19
2.4.3 Algorithm . . . . . 20

**3 State of the Art** **23**

**4 Methodology** **27**
4.1 Data Description . . . . . 27
4.2 Cell Recognition Model . . . . . 28
4.2.1 Data Preparation . . . . . 28
4.2.2 Convolutional Neuronal Network Architecture . . . . . 29
4.2.3 Training . . . . . 30
4.3 Environment and Agent Design . . . . . 30
4.3.1 Agent . . . . . 30
4.3.2 Environment . . . . . 31
4.3.3 Resetting the Environment . . . . . 33
4.3.4 Goal and Reward Signal . . . . . 33
4.4 Implementation . . . . . 35
4.4.1 Software . . . . . 35
4.4.2 Hardware . . . . . 35
4.4.3 Pseudocode . . . . . 35
4.5 Performance . . . . . 36
4.6 Experiments . . . . . 37

**5 Results and Discussion** **39**

5.1 Training of the Agents . . . . . 39

5.1.1 Firs Stage . . . . . 39

5.1.2 Second Stage . . . . . 43

5.1.3 Third Stage . . . . . 44

5.2 Agents Behavior From Manually Testing . . . . . 44

5.2.1 First Stage . . . . . 45

5.2.2 Second Stage . . . . . 47

5.2.3 Third Stage . . . . . 50

5.3 Training Method . . . . . 54

5.3.1 Sample Efficiency . . . . . 54

5.3.2 Problem Difficulty . . . . . 55

5.3.3 Neuronal Network . . . . . 55

5.3.4 Performance Metric and Reward Signal . . . . . 56

5.3.5 Reliability and Validity . . . . . 56

5.3.6 Source Criticism . . . . . 57

5.4 Future Works . . . . . 57

**6 Conclusions** **59**

**Bibliography** **61**

# List of Tables


4.1 Summary table of the feature extractor in the agent's CNN architecture. . 31

4.2 Summary table of the dense layers in the agent's CNN architecture. . . . . 31

4.3 Agent's actions. . . . . 32

4.4 The hyperparameters included in the PPO Stable Baselines 3 implementation, with descriptions and values. The symbol following a hyperparameter name refers to the coefficient. . . . . 37

4.5 Brief description of the agents. . . . . 38

# List of Figures


2.1 Interaction agent-environment: The agent execute an action $A_t$ in the environment. The environment returns a current state $S_t$ and gives reward $R_t$ to the agent. The agent perform another action taking into account the information and reward obtained from the environment. . . . . 6

2.2 The values in each box are calculated by using the state-value function $v_\pi(s)$. The agent moves through the grid-world environment collecting the rewards until reaching the goal box or receiving a penalty for falling into the box with the skull. . . . . 9

2.3 Graphic representation of a neuron. Given $x_n$ inputs, they are multiply by a corresponding weight and summed up. The result of this operation is passed through an activation function which generates the neurons output. . . . . 12

2.4 Graphic representation of a deep feedforward network. The network is built by columns known as layers where each node is a neuron. . . . . 13

2.5 Graphic representation of a $2 \times 2$ kernel . . . . . 14

2.6 Convolution operation of a $4 \times 4$ matrix with a $2 \times 2$ kernel . . . . . 14

2.7 Convolution operation of a $6 \times 6$ matrix with padding with a $2 \times 2$ kernel 15

2.8 Graphic representation of the mathematical operation stride. The image shows a stride of 3. . . . . 15

2.9 Graphic representation of Max pooling and Average pooling . . . . . 16

2.10 Standard neuronal network and neuronal network with dropout . . . . . 16

2.11 Graphic representation of the data augmentation process. . . . . 17

2.12 Graph shows a single timestep of the surrogate function, $L^{CLIP}$. The left graph shows the clipping for positive advantages and the right graph shows the clipping for negative advantages. Besides, if $A = 0$ then $L^{CLIP} = 0$ . . 19
2.13 The graph represent a shared network. The first layers are shared by the policy and value function. However, the last layers, known as heads, are split in specific layers for the policy/value function. . . . . 20

4.1 Three samples of the dataset retrieved from Mendeley data repository . . 28
4.2 The right hand samples shows two cropped images with cells. The left hand samples shows two images of the background of tests . . . . . 28
4.3 Set of images with cells used for training the cell recognition model . . . . 29
4.4 Set of images of background without cells from samples. . . . . 29
4.5 Graphic representation of the data flow through the CNN architecture. . . 30
4.6 Graphic representation of two environments. The environment has two different parts, the A part shows the searching windows where the agent is moving and what the ROI capture is shown in the right window. On the other hand, The B part shows the detected cells bounded by a black square, and the detected cell is represented, in more detail, in the left window. . . 32
4.7 Graphic representation of the observation space. The right hand side figure shows the current state $s_t$, while the middle. $s_{t-1}$ and the left hand side, $s_{t-2}$, figures shows the previous states . . . . . 33
4.8 Interaction between the agent, cell recognition CNN and the environment . 34
4.9 Different situations when the agent receive a reward or penalty. . . . . 34
4.10 Graphic representation of the experiments. . . . . 38

5.1 The figure shows the first four agents using the hyperparameter n_steps equal to 512. The plots represent the obtained score at the end of each episode. The $x$-axis shows the time steps, until 3 million steps. The $y$-axis shows the score obtained in the episodes. . . . . 40

5.2 The figure shows the second set of agents which use the hyperparameter n_steps equal to 1024. The plots represent the obtained score at the end of each episode. The $x$-axis shows the time steps, until 3 million steps. The $y$-axis shows the score obtained in the episodes. . . . . 41
5.3 The right hand side plot shows all the first set of agents scores, permitting the ability to visually compare better their training process. The left hand side plot shows only three agents, the agent D was not considered because it has lowest values. . . . . 42
5.4 The right hand side plot shows all the second set of agents scores, permitting the ability to visually compare better their training process. The left hand side plot shows only three agents, the agent H was not considered because it has lowest values. . . . . 42
5.5 The graph shows the retraining results of the agents C, E and F which showed good results and searching behaviors. The left bottom figure shows the merged results of the agents for a better visualization of the training process. . . . . 43
5.6 The graph shows the retraining results of the agents C, E and F which showed good results and searching behaviors in the previous stage. . . . . 44
5.7 The figure shows how the untrained agent perform random actions and it has a frenetic behavior. The red dots are cells that the agent found during the episode. The blue bar indicates how many times the agent pass trough the same position. . . . . 47
5.8 The figure shows three different environments which were used for testing the agents in the second stage and third stage. . . . . 48
5.9 The figure shows the tracking of the agent C, while searching in the three mentioned environments. . . . . 48
5.10 The figure shows the tracking of the agent E, while searching in the three mentioned environments. . . . . 49
5.11 The figure shows the tracking of the agent F, while searching in the three mentioned environments. . . . . 50

5.12 The figure shows the tracking of the agent E1, while searching in the three mentioned environments. . . . . 51
5.13 The figure shows the tracking of the agent E2, while searching in the three mentioned environments. . . . . 52
5.14 The figure shows the tracking of the agent E3, while searching in the three mentioned environments. . . . . 53
5.15 The figure shows the tracking of the agent E4, while searching in the three mentioned environments. . . . . 54

# Chapter 1

# Introduction

## 1.1 Background

The concept of reinforcement learning (RL) in computing sciences is inspired from biological behaviors. In order to clarify it, Ellis Ormrod in her book *Human learning* [1] defines *learning* as a consequence of associating results from long-term experiences which induce a change in mental representation [1]. Consciously or unconsciously, we are applying this concept into practice daily, and a good example could be when we develop new behaviors or how rewarding a dog with food makes it learn a trick. In addition, the connection between reward and a new behavior is called *reinforcement* [1] in other words, cause and effect consequence.

Reinforcement learning, which is a machine learning area, arise from the curiosity of create intelligent agents through a trial and error process. The agents learns from stimuli known as reward, that is given depending on the interaction of the agent inside an environment, it could be a good action or a bad action. The aim of an learner is to maximize a numerical reward signal while discover what actions gives a maximum reward in each step of an episode. Finally, the agent learns good behaviors in the environment [2].

One of the most important advantage of RL is that the developer does not have to train the agent personally because the learning is based on the reward signals [2]. The only task of the developer is to define the rewards on the environment. This task is not easy, and the environment must be built taking into account what the agent need to learn and what the agent has to do at the end of his training.

## 1.2 Problem Statement

Cervical cancer is one of the most common and deadly cancers in women, approximately more than five hundred million of women are diagnosed with cervical cancer per year, and it cause over than three hundred million of deaths around the world [3, 4]. Almost, one hundred percent of cervical cancer has connection with infection of human papillomaviruses (HPV), a common virus that is transmitted by sexual relations [4]. The prevention of the cervical cancer is widely possible, because there is a treatment which consist firstly in HPV vaccination and secondly prevention approaches such as screening for, and treating cervical lesions [5]. In case of screening for cervical cancer there are two kind of tests. Papanicolaou test commonly know as Pap smear test, and HPV test. The HPV test focus on detect the HPV virus existence, while Pap smear test collects cells from cervix and looks for affected cells which may cause cervical cancer if it is not treated correctly. The professional recommendation is that Pap test may be performed in women who are over 21 years old, and the frequency is determined by the doctor [5].

HPV vaccine, HPV test and Papanicolaou test have high efficiency for preventing cervical cancer. However, these treatments do not show good results in low-income and middle-income countries where the ninety percent of cervical cancer occur. In this countries, the vast majority of women have no chance of getting an HPV vaccine or Pap smear screening, what increases the risk of cervical cancer which is over ninety percent [3, 6, 7]. Without surprise, in Latin America and Caribbean countries, low-income and middle-income countries, there exist social, economical and cultural barriers that do not allow women participate in HPV vaccine or screen for programs [6, 8]. By 2020, in Ecuador only thirty six percent of women has had covered with the HPV vaccination program, and by 2019 five in ten women have been screened for cervical cancer in the last 5 years [9]. Furthermore, in Ecuador close to 50 percent of women over 18 years old have never done a cervix screening [10]. Currently, statistics shows that there exist a low/medium coverage in screening for cervical cancer persists. This is caused by the mentioned problems that suffer low-income and medium-income countries.

Healthy system barriers: shortage of personnel, deficient health service, long waiting times, lack of adequate instruments and equipment, shortage of medical supplies and many

others [6].

Cultural barriers: lack of sexual education in terms of preventive methods and use of condoms, high rates of promiscuity in young people and sex is still a taboo in those countries [6].

Knowledge barriers: lack of information about benefits of Pap smear test or HPV tests and cervical cancer preventive treatments [6].

This thesis focused on healthy system barriers, precisely in the long waiting times that patients have to do in order to get their test results. A poll carried out to 81 women from urban and rural areas of Ecuador in 2009 showed up the dissatisfaction for the long waiting times in order to received results of their tests. The retardment comes from the lack of adequate instrument, limited capacity of trained personnel and specialist, usually in rural areas [11]. The analysis of cervix screening can be only carried out by three kind of specialist such as: anatomic pathologists, cytologist and clinical pathologists. This reduced amount of specialties for evaluating Pap smear test increase the difficulty to evaluate more tests, according to a report of WHO executed in Ecuador in 2021 which shows that the number of medical staff per 10000 cancer patients: zero radiation oncologists (2019), 3 medical physicists (2019), 154 radiologists (2019), and 4 nuclear medicine physicians (2019) [9]. For this reason, in order to delete the long waiting times, the use of new technologies and artificial intelligence is necessary. The aim of the thesis is to implement a deep reinforcement learning (DRL) technique for detecting nucleus cell location in digital Pap smears tests. Therefore, the main contribution of the thesis is to speed up the evaluation of Pap smear in order to reduce a healthy system barrier.

## 1.3 Objectives

### 1.3.1 General Objective

Develop a DRL agent for cells recognition in digital Pap smears in order to reduce long waiting times for Pap smear diagnostic.

### 1.3.2 Specific Objectives

- Generate a deep reinforcement learning environment to train an agent with different digital Pap smear.
- Evaluate the effectiveness of the environment training multiple agents with the Proximal Policy Optimization (PPO) algorithm.
- Modify the environment parameters to test if the agents can be trained more efficiently and converge faster to an expected behavior.

# Chapter 2

# Theoretical Framework

This chapter covers an overview of concepts in the field of reinforcement learning as well as the PPO algorithm. In addition, relevant concepts of neuronal networks, convolutional neuronal networks (CNN) are presented.

## 2.1 Reinforcement Learning

Reinforcement learning is a branch of machine learning that focus on create intelligent agents during a trial and error learning inside of an environment. The environment provide information about the actions that agent can take, returns penalties or rewards and the observation space. The aim of an agent is to maximize the amount of rewards winning along the training episode, given a feeling of satisfaction. Finally, at the end of this process of learning the agent can figure out how to solve the problem that was given [12].

### 2.1.1 Environment

The environment is the space where the agent moves around and learn from it. The environment can not be modify by the agent, the environment only allows to the agent to take the defined actions. Furthermore, the interaction that exist between the agent and environment in reinforcement learning can be shown as the Fig. 2.1.

This interaction is a sequence of discrete time steps $t = 0, 1, 2, \ldots, n$. By each time steps the environment return a current state $S_t$, an action $A_t$, a reward $R_{t+1}$ and a observation $\Omega$ [2]. The environment could be either partially observable or fully observable. In case of a partially observable environment the agent is unaware about the exact state of the whole.

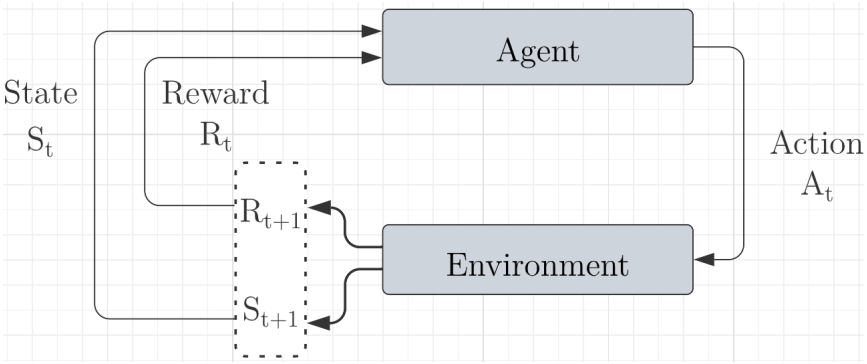



Figure 2.1: Interaction agent-environment: The agent execute an action $A_t$ in the environment. The environment returns a current state $S_t$ and gives reward $R_t$ to the agent. The agent perform another action taking into account the information and reward obtained from the environment.

In contrast, in a fully observable environment the agent is well informed about the stat it is in [13]. Formally, an environment in reinforcement learning is defined as a partially observable Markov decision processes which is described as a 6-tuple, see Eq. 2.1.

$$\mathcal{P} = (S, A, T, R, \Omega, \gamma) \tag{2.1}$$

Where,

- $S$ is the set of states, $\{S_1, S_2, \ldots, S_t\}$.
- $A$ is the set of actions, $\{a_1, a_2, \ldots, a_n\}$.
- $T$ is the set of conditional transition probabilities, $T(s', s, a)$.
- $\Omega$ is the observation space.
- $R$ is the reward function.
- $\gamma$ is the discount factor, $\in [0, 1]$.

### 2.1.2 Agent

The agent is the controller, decision-maker, actor and learner in the process of learning in reinforcement learning. The aim of the agent is to perform actions that interact with the environment and maximize the numerical reward. Each episode in the environment

is conformed by a sequence of steps $t = 0, 1, 2, \ldots, T$, after $T$ steps the episode ends and starts a new episode. The steps are the actions that the agent makes, and in each of those steps the environment gives information to the agent about the state and reward. The agent store that information and try to perform steps in order to obtain a higher reward at the end of an episode. The interaction behavior can be seen in the Figure 2.1, [2].

### 2.1.3 Policy

The agent may preform better results as well as the training process progresses. This behavior is possible by choosing the right action in each step. The only way to achieve the expected behavior is using a policy $\pi$. Basically, the policy in reinforcement learning is what allows us to make intelligent agents. The policy use the information retrieved from the environment such as the state and action for mapping the best action in each time step. Depending on the task that agent have to figure out, the policy could be a function, a register or in other cases it may be a hard computational process. Furthermore, the policy is usually stochastic because $\pi(a|s)$ represent the probability for taking an action $a$ in the current state $s$ [2].

### 2.1.4 Reward Signal

In reinforcement learning the reward signal $R_t$ defines the goal achieved by the agent during the interaction environment-agent. In each time step $t = 0, 1, 2, \ldots, T$ the agent receive a single number known as reward. The agent focus on maximize the score in each time step; the score, also called expected return $G_t$, is the sum of all the rewards obtained in the training process and it is represented as the equation, see Eq. 2.2.

$$G_t = R_{t+1} + R_{t+2} + \ldots + R_T = \sum_{k=0}^{T-t-1} R_{t+1+k} \tag{2.2}$$

In some cases, the return has implemented a discount rate $\gamma$ which suppress the result of future rewards in order to improve the learning process. The discount rate factor is applied when the environment has not a final time step, $T = \infty$, and the return reward likely could be infinite. Using the discount factor the equation remains as follows, see Eq.

2.3

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \tag{2.3}$$

where the discount rate is bounded as $0 \leq \gamma \leq 1$, [2].

### 2.1.5 Value Function

The value function estimate the performance of an action in a state. The value is defined by the expected future rewards in a state and under a policy. The way of measure the value function is taking into account the current state, the actions that can be performed, the discount rate factor and the policy. Commonly, the value function could be defined by using the policy $\pi(a|s)$, where the $a$ is the action taken in a state $s$. Hence, the value under a policy is denoted as $v_\pi(s)$. In case of finite Markov decision process states the equation can be written as follow, see Eq. 2.4

$$v_\pi(s) = \mathbb{E}_\pi \left\{ R_t \mid s_t = s \right\} = \mathbb{E}_\pi \left\{ \left( \sum_{k=0}^{T-t-1} \gamma^k R_{t+1+k} \right) \mid s_t = s \right\} \tag{2.4}$$

where $\mathbb{E}$ is the statistical estimation for future expected rewards. Furthermore, the function 2.4 is know as the state-value function for policy $\pi$, [2, 12].

The Fig. 2.2 shows how the agent, a robot, is in a grid-world environment where he may take actions as up, left, right and down in order to reach the goal square. Furthermore, in the environment, each square shows a value which is the state-value function for that state. The state-value function is higher while the squares are closed to the goal square and lower when the square is closed to the pit.

### 2.1.6 Episode

An episode or sometimes called trials is a sequence of time steps until the agent reach the goal or he gets special state know as *terminalstate*. During an episode the agent and the environment interact as the Fig. 2.1. After determined $t$ time steps the episode ends and the environment is reset, including the time steps, the score and the agent is located in the starting state. As is shown in Fig. 2.2, in the grid-world environment the each step is a move that the robot performed until he reaches the goal square or when the episode

conclude in a terminal state [2].

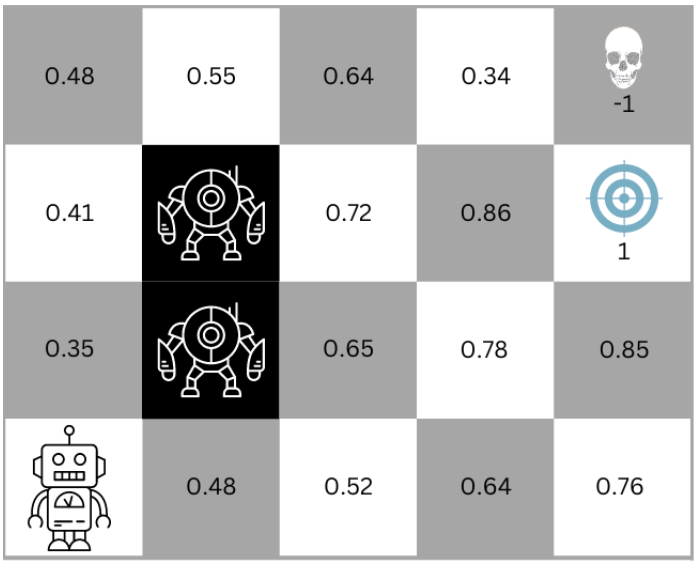



Figure 2.2: The values in each box are calculated by using the state-value function $v_\pi(s)$. The agent moves through the grid-world environment collecting the rewards until reaching the goal box or receiving a penalty for falling into the box with the skull.

### 2.1.7 Exploitation and Exploration

In order to achieve good rewards the agent explores the environment looking for the goal. The dilemma arises when the agent has an effective way for getting good rewards and he exploit what it knows, but in the environment could be exist better ways to get higher scores. For this reason, the agent has to be forced to explore the environment for achieving new solutions. In reinforcement learning is commonly used the $\epsilon$-greedy which is a exploring policy that force the agent for searching new solutions. The $\epsilon$ is the probability for the agent to perform a random action and a $1-\epsilon$ probability for the agent to choose an action based on the previous knowledge [2, 12].

### 2.1.8 *Q*-learning

In reinforcement learning $Q$-learning is an algorithm that in the same way of state-value function 2.4 which try to determine how good is a state, the $Q$-learning algorithm consider either the action-value function $Q_\pi(s, a)$ and the state-value function $v_\pi(s)$ in order to determine how good is an action $a$ in a state $s$. The action-value function is represented

in the equation Eq. 2.5.

$$Q_\pi(s,a) = \mathbb{E}_\pi \{r_t \mid s_t = s, a_t = a\} = \mathbb{E}_\pi \left\{ \left( \sum_{k=0}^{T-t-1} \gamma^k r_{t+1+k} \right) \mid s_t = s, a_t = a \right\} \tag{2.5}$$

The $Q$-learning algorithm uses an $\epsilon$-greedy policy where $\epsilon$ is the probability of take a random action and a $1-\epsilon$ probability of perform an optimal action which is the action that maximizes the action-value function $\max_a Q_\pi(s,a)$ [2, 14].

Commonly, the $Q$-learning algorithm is initialized using arbitrary random values. After some interaction between agent-environment the chosen action by the agent are based on the agent's policy. Thereafter, the action-value function is updated following the next equation, see Eq. 2.6.

$$Q_\pi(S_t, A_t) \leftarrow Q_\pi(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q_\pi(S_{t+1}, a) - Q_\pi(S_t, A_t) \right] \tag{2.6}$$

The learning rate $\alpha$ determines how the old value is affected by the update. This behavior occurs during each episode until it finished. Thereafter, the environment is rest and all starts again [2]. The $Q$-learning pseudo-code which shows the values stream in the process of learning can be seen in the algorithm 1.

**Algorithm 1:** $Q$-learning

```
Initialize Q_π(s, a) with random values;
for episode do
    Reset environment;
    while not done do
        Choose action A based on state S using policy π;
        Perform action A and observe R, S';
        Q_π(S, A) ← Q_π(S, A) + α [R + γ max_a Q_π(S', a) − Q_π(S, A)] ;
        S ⟵ S';
        if episode end then
            done;
        end
    end
end
```

## 2.2 $Q$-function Approximation

Depending on the task that the agent has to figure out it can be apply different function approximation techniques. In some cases, the action-value and state-value function are tables which storage all possible state and actions. This table storage technique is apply when the amount of states are small, look the Fig. 2.2 where the environment is a small grid-world with 18 states, but in the real world there are environments with lager state space. In this graduate project images are used, it becomes the state space too big for being storage in a table. Calculating a state table from an image that is $250 \times 250$ pixels large, each pixel has a RGB value representation in the range of $0 - 255$. The state space has $256^{3(250\times250)} \approx 10 \times 10^{451544}$ possible states which is a number of possible states that can not be fitted in a table. Using the $Q$-learning algorithm for solving this task the computational time and computational resource is unbelievable expensive. For this kind of task where images are used, there are other function approximation techniques, such as neuronal networks, which can deal with large state tables and learn to approximate a state-value function [2, 12, 13, 15].

### 2.2.1 Feedforward Neuronal Networks

Feedforward neuronal networks is another function approximation technique. The aim of this technique is to approximate a given function $y = f^*(x)$ by mapping an input $x$ to the output $y$. The feedforward neuronal networks maps $y = f(x;\theta)$ and learns from the parameters $\theta$, commonly know as the weights of the network, in order to get the best function approximation [13, 15].

The meaning of feedforward in this model is because the information from $x$ flows into the evaluated function, in the intermediate computations which define $f$ and finally the output $y$. Furthermore, feedforward neuronal networks use the word networks because the model is represented as the composition of many different functions. For instance, there are three functions $f^1$, $f^2$, and $f^3$ which are connected in the usually structure, a chain, to form $f(x) = f^3(f^2(f^1(x)))$. The first function $f^1$ is called first layer, $f^2$ is called the second layer and $f^3$ is called the third layer, the set of this layer is called hidden layers, see Fig. 2.4 [15]. Finally, in neuronal networks the main element is the neuron, see Fig. 2.3. Multiply

the neuron's inputs $x$ by the weights $\theta$ and sum them. This becomes the neuron's output, which is sent to the activation function. The role of the activation function is to add the non-linearity necessary for the network to approximate an arbitrary function. There are various activation functions, ReLU (Rectified Linear Unit) being the most commonly used in modern neural networks. This activation function is defined as follow, see Eq. 2.7.

$$g(z) = \max(z, 0) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

The summed output of the neuron is $z$ [15]. A neuronal network is obtained after connect layers of neurons, see Fig. 2.4. Such networks have the ability to approximate nonlinear functions if the weights are properly adjusted. For example, it can be used to approximate an agent's value function or policy. Where in that case, the input of the network is the state $s$.

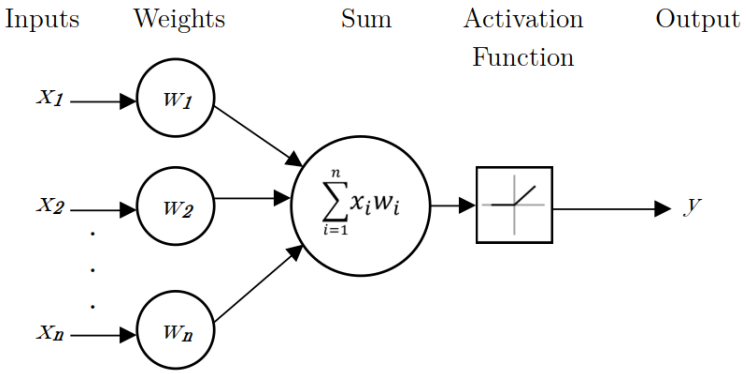



Figure 2.3: Graphic representation of a neuron. Given $x_n$ inputs, they are multiply by a corresponding weight and summed up. The result of this operation is passed through an activation function which generates the neurons output.

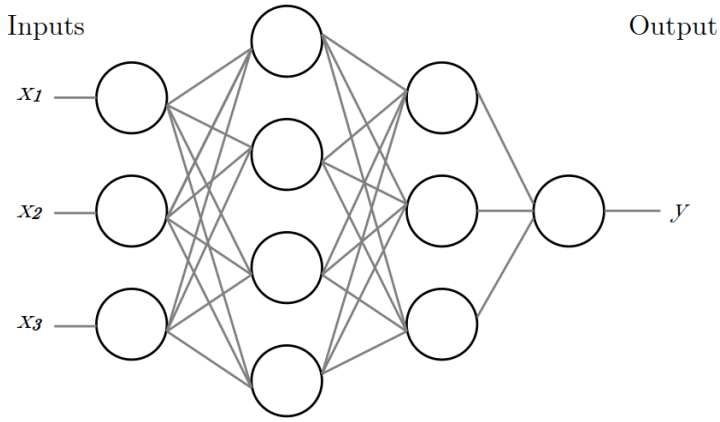



Figure 2.4: Graphic representation of a deep feedforward network. The network is built by columns known as layers where each node is a neuron.

### 2.2.2 Convolutional Neuronal Network

Convolutional neuronal networks (CNN) are computer processing systems inspired by biological nervous systems (similar to the human brain). In the system, a neuron receive a stimulus and process that input to execute some operation [16]. CNN are widely used in machine learning techniques because they are capable to deal with high-dimension data such as images or videos. A CNN is structured by a set of convolutional layers. The first convolutional layer receive an input (image), it apply a filter to the image for making convolutional operations that extract features of the image. The result of this convolution is a new image which is fed to the next convolutional layer and so on [13]. Finally, the extracted features are used, as example, for detecting objects in the image or for image classification. There are some remaining topics of CNN which are treated more details in the next sections.

#### Filter

A filter or commonly called kernel is a matrix full of values called weights, see Fig. 2.5. The kernel swept over the image and multiplied the input to enhance the output in a particular and desirable way. During the process of training the kernel is update in order to correctly extract the desirable features [17].


Figure 2.5: Graphic representation of a $2 \times 2$ kernel

### Convolutional Operation

The convolution operation is a simple multiplication between the filter and the input, both may be of the same dimension. The input enter in the CNN and the neurons in each layer perform this operation. The input of the next convolutional layer is the result of the previous one, usually the result has smaller dimensions than the input. In the case of this project, the input is an image RGB, which is an arrangement of pixels [17].

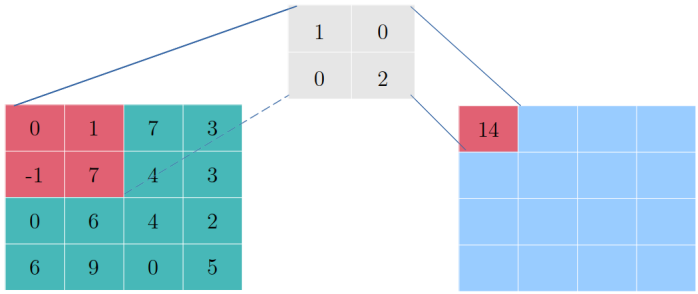



Figure 2.6: Convolution operation of a $4 \times 4$ matrix with a $2 \times 2$ kernel

### Padding

The convolutional operation has a border problem because it tends to loos the border pixels. In the Fig. 2.6 where the kernel is a matrix $2 \times 2$, it is easy to see that the pixel of the first column is only evaluated one time during the convolution, while the pixels in the middle of the input matrix are evaluated at least 3 times. This cause loos information from the edges of the input matrix. Apparently, it is not a big problem, but in fact when multiple convolutional layers are apply the amount of information that is loosing is considerable and some important features can not be detected. Since, we known that information from the bounds can be lost during the convolution, to overcome this it is introduce Padding to the image. Padding is a technique that extend the are of the input matrix adding an extra layer to the matrix. Usually, the extra layer is filled with zeros and it is called the zero-padding, see Fig. 2.7 [18].

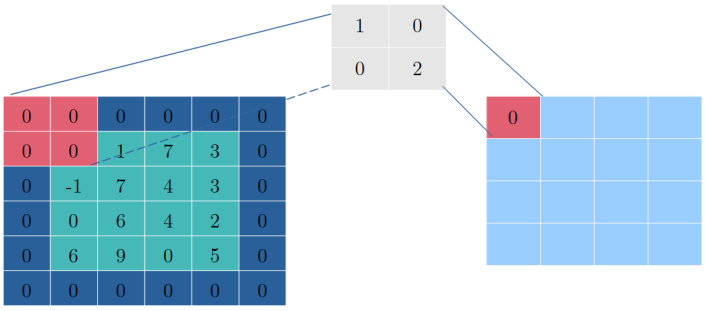



Figure 2.7: Convolution operation of a $6 \times 6$ matrix with padding with a $2 \times 2$ kernel

### Strides

Convolutional neuronal networks commonly include multiple downsampling operators, such as strided convolutions that gradually reduce the resolution of input representation. The strides represents the distance that the kernels moves from a state to other state. The stride could be one which evaluate the kernel each step. However, it is not necessary that amount of information, that is why it could be considered other values values in order to extract all the important information reducing the computation cost [19].

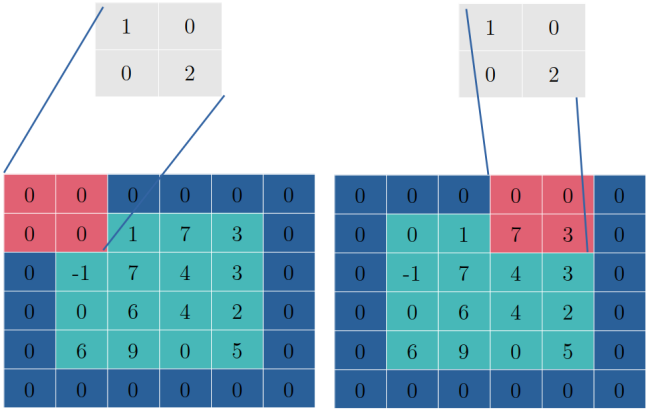



Figure 2.8: Graphic representation of the mathematical operation stride. The image shows a stride of 3.

### Pooling

A CNN consists of alternating convolutional and pooling layers. Pooling technique apply a pooling operator to add information in each small range of the input feature channel and downsampling the result. This technique reduce the output from the previous convolutional layers in order to get a compact result but keeping important information. There are several

pooling techniques, the most common is Max pooling which extract the highest value from the evaluated area. On the other hand, Average pooling extract the average value from the evaluated area [20].

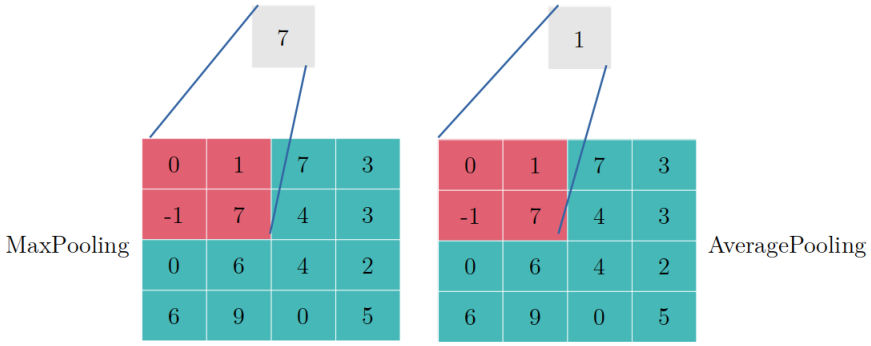



Figure 2.9: Graphic representation of Max pooling and Average pooling

### Dropout

Dropout, as its name suggests, refers to eliminating values from a neuronal network reducing the danger of over-fitting and indirectly investigating new architectural models. This technique is a stochastic process used in CNN where different neurons are deactivated during the training, the discarded neurons are selected at random by a probability value. The dropout probability must to be carefully adjusted since a high dropout incidence removes too much data, making it difficult for the CNN to learn the image's features and causing under-fitting [21, 22].

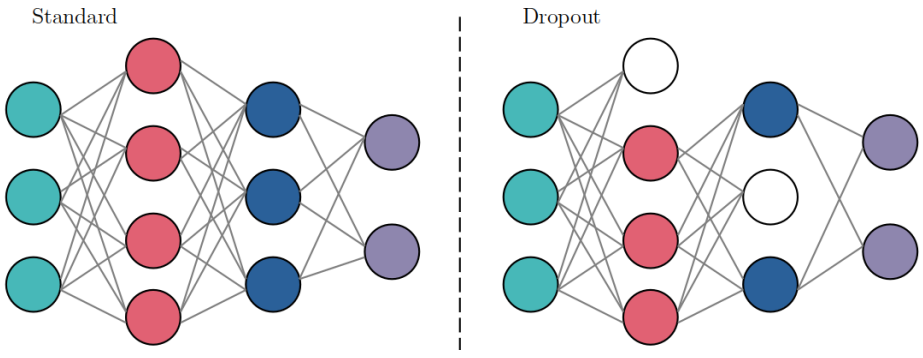



Figure 2.10: Standard neuronal network and neuronal network with dropout

## 2.3 Data Augmentation

Data augmentation is a method to provide more diversity to the training set by rotating, flipping, cropping, zooming the images or implementing other random (yet realistic) changes. It serves as a regularizer and aids in lowering over-fitting while a model is being trained [23].

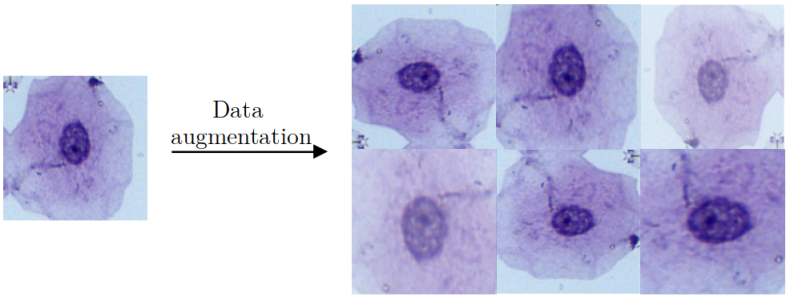



Figure 2.11: Graphic representation of the data augmentation process.

## 2.4 Proximal Policy Optimization Clip

PPO is an algorithm of the family of policy gradients methods, introduced by OpenAI, for training models in reinforcement learning. It is an algorithm that collect all the best form other algorithms such as trust region policy optimization thus ensuring facility of code and tune, training stability, avoid having large policy updates, etc. In recent years, PPO has become one of the most used algorithms to build agents for playing complex games such as video games, table games or even solving human tasks such as automatic driving [24, 25].

This section examines the techniques on which PPO is base, policy gradient methods, before describing the clipping that is used to prevent massive policy updates. Finally, the algorithm is provided.

### 2.4.1 Policy Gradient

A reinforcement learning agent's goal is to maximize the "expected" reward in a state $s$ after an action $a$ while obeying a policy $\pi$. Policy gradient techniques are used to learn a parametrized policy $\pi(a|s,\theta)$ where $\theta$ is the set of parameters which could be the weights from a given neuronal network. Furthermore, instead of using a deterministic policy $\pi(s|a)$

it is used a parameterized policy $\pi$, using the parameters $\theta$, for yield a probability of performing an action $a$ in a specific state $s$ at time step $t$, see Eq. 2.8.

$$\pi(a \mid s, \theta) = \mathrm{P}\left\{A_t = a \mid S_t = s, \theta_t = \theta\right\} \tag{2.8}$$

Policy gradient approaches seek to optimize a performance measure $J(\theta)$ [2], often known as objective or loss [24]. The updates to $\theta$ are carried out using gradient ascent, see Eq. 2.9.

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J\left(\theta_t\right)} \tag{2.9}$$

where $\widehat{\nabla J\left(\theta_t\right)}$ is the deriving the policy gradient, or the performance measure's gradient [2]. There are several policy approximation methods, but PPO uses policy gradient methods, commonly know as actor-critic. Actor-critic in addition to learn from the values of states or state-actions pairs, it also takes into account and learns from the policy. PPO algorithm bases his gradient estimator in vanilla policy objective which has the form, see Eq. 2.10.

$$\widehat{\nabla J\left(\theta_t\right)} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) \hat{A}_t\right] \tag{2.10}$$

where $\mathbb{E}$ is the expectation which means the empirical average of a limited batch of samples. $\hat{A}_t$ is a time step $t$ estimator of the advantage function which is expressed as follow, see Eq. 2.11.

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{2.11}$$

where,

$$\delta_t = r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{2.12}$$

and $\lambda$ is a coefficient. Using the gradient estimator, 2.10, the policy gradient loss function has the form, see Eq. 2.13 [24].

$$L^{\mathrm{PG}}(\theta) = \widehat{J(\theta)} = \hat{E}_t\left[\log \pi_\theta\left(a_t \mid s_t\right) \hat{A}_t\right] \tag{2.13}$$

### 2.4.2 Clipping

The idea behind PPO is to confine our policy update using a new objective function called the clipped objective function, which uses a clip to restrict the policy change in a tiny area. On the other hand, policy gradient loss, $L^{PG}(\theta)$, do not use a clip what cause that the huge amount of updates, in many cases in the same trajectory, destroy and corrupt the right behavior of the policy. Hence, this new function is intended to avoid damaging huge parameters updates, see Eq. 2.14.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[ \min \left( r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{2.14}$$

where $\epsilon$ is the clipping range, commonly $\epsilon = 0.1$ or $\epsilon = 0.2$ and $r_t(\theta)$ is the ratio function defined as follow, see Eq. 2.15.

$$r_t(\theta) = \frac{\pi_\theta \left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t\right)} \tag{2.15}$$

By clipping $r_t(\theta)$ ensures that the probability ratio does not exceed the boundaries. Hence, the clip function ensure that the current policy cannot be too distinct from the previous one [24].

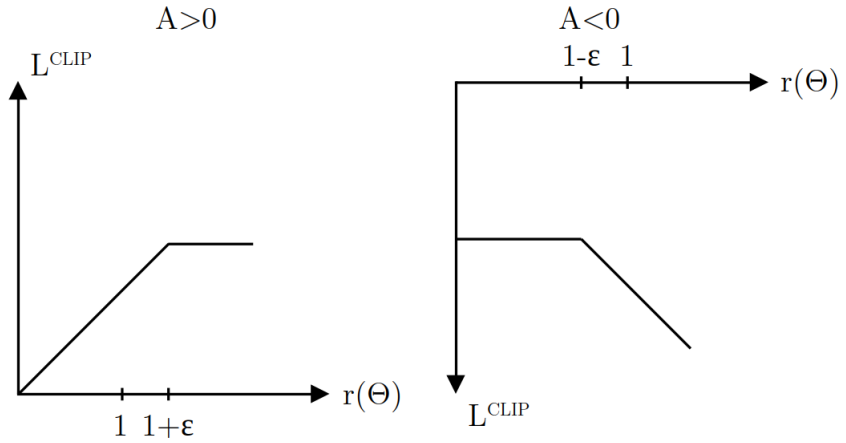



Figure 2.12: Graph shows a single timestep of the surrogate function, $L^{CLIP}$. The left graph shows the clipping for positive advantages and the right graph shows the clipping for negative advantages. Besides, if $A = 0$ then $L^{CLIP} = 0$

### 2.4.3 Algorithm

In case when the policy and value function are approximated using neuronal networks it is necessary utilize a shared network rather than two distinct networks. In shared networks the first layers are shared, which means that they are utilized for policy and value approximations. On the other hand, the last layers are separated into distinct layers that are customized to the specific approximation, see Fig. 2.13.

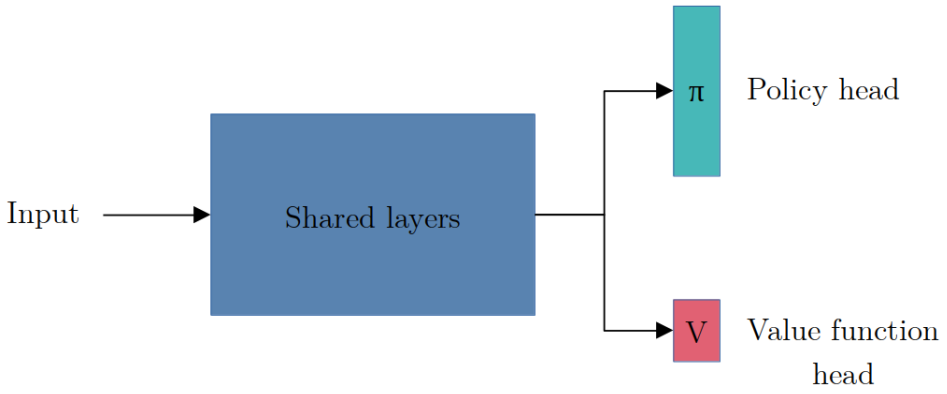



Figure 2.13: The graph represent a shared network. The first layers are shared by the policy and value function. However, the last layers, known as heads, are split in specific layers for the policy/value function.

When using a shared network, the goal should be a merge of $L^{CLIP}(\theta)$ and value function loss which has the form, see Eq. 2.16.

$$L^{\mathrm{VF}} = \left(V_{\boldsymbol{\theta}}\left(s_t\right) - V_t^{\mathrm{target}}\right)^2 \tag{2.16}$$

In addition, a term called *Shannon* entropy $S_{\pi_\theta}(s_t)$ is needed and is also applied in the loss function [24]. The entropy ensures that the algorithm explores and not just exploits what it knows, see Eq. 2.17.

$$S_{\pi_\theta}\left(s_t\right) = -\sum_{i=1}^{n} \pi_\theta\left(a_i \mid s_t\right) \log\left(\pi_\theta\left(a_i \mid s_t\right)\right) \tag{2.17}$$

Where the probability of taking an action $a$ in a specific state $s$ and following a policy $\pi_\theta$ is determined by $\pi_\theta(a|s)$ [26]. Merging these important terms, the function looks, see Eq. 2.18.

$$L_t^{\mathrm{CLIP+VF+S}}(\theta) = \hat{E}_t\left[L_t^{\mathrm{CLIP}}(\theta) - c_1 L_t^{\mathrm{VF}}(\theta) + c_2 S_{\pi_\theta}\left(s_t\right)\right] \tag{2.18}$$

In the equation, see Eq. 2.18, $c_1$ and $c_2$ are coefficients. The first coefficient $c_1$ is know as the value function coefficients and secondly $c_2$ is the entropy coefficient.

The PPO algorithm, see algorithm 2, starts by initializing the parameters or in this case the weights of the CNN $\theta$. Then, at each time step T an action is carried out following the $\pi_{\theta,old}$ policy, thus creating what is known as the trajectory. The advantages are computed, and when the data is gathered, the policy is changed by optimizing $L_t^{\text{CLIP+VF+S}}(\theta)$ with respect to $\theta$. Using the proposed policy gradient, the optimization is carried out multiple times for the same trajectory. The epochs are defined by the optimizations steps in the same trajectory [24].

**Algorithm 2:** PPO Clip

```
Initialize θ;
for iteration i = 0, 1, ... do
    for time step t = 0, 1, ..., T do
        Sample time step with policy π_{θ,old};
        Calculate advantage Â_t;
    end
    for epoch k = 0, 1, ..., K do
        Optimize L^{CLIP+VF+S} with respect to θ;
        Update θ;
    end
end
```

# Chapter 3

# State of the Art

In reinforcement learning, the environment is a fundamental part because it is where an agent interact and learn of it. Remark, in RL an environment is formally defined as a partially observable Markov decision process [2]. An episode in the environment is conformed by steps which are the actions that the agent takes, if the agent do a correct movement it will obtain rewards that after many episodes and applying Q-Learning algorithm the agent will learn what to do in the environment. Taking into account it, Brockman et al. designed OpenAIGym which is a toolkit for RL research. This toolkit compile a variety of different environments, including: Classic control and toy text, algorithmic, Atari, board games and 2D and 3D robots. Furthermore, using OpenAIGym it is possible to build our own customized environment [27].

DeepMind Technologies team were the first who presented a deep learning model which effectively learn control polices from Atari games using reinforcement learning. Researchers pass the Atari games frames as raw pixels through a CNN in order to extract features of the image and build an intelligent agent. Firstly, the paper shows that CNN overcome the challenge of learn control policies from video data using a similar approach of Q-learning. Secondly, the output of the CNN is an estimated future rewards from a function which represent the learning factor. Finally, using reinforcement learning to learn how to play Atari games resulted in an agent that can play better than the average of humans [28].

Desai and Banearjee [29] proposed a project of learning of play Space Invader, an Atari game. The objective was an agent that interact with the environment and learn about passed experiences and action-reward behavior that are stored in a Replay Memory. They

implemented improved algorithms like Double Q-Learning which is an extension of Q-Learning, including better architectures for learning. Besides, they worked with raw pixels and RAM state representation which is emulator's RAM chain. After training and test episodes, the main conclusions were: the use of Dropout affect the agent capability of learning, and RAM representation requires low computational power unlike high dimensional representation.

Moreno in his paper "Performing Deep Recurrent Double Q-Learning for Atari Games" shows an improvement for the algorithms Double Q-Learning and Recurrent Networks. The architecture used in the implementation of the DQRN has a first network with 3 convolutional 2D layers and 2 fully connected layers, and the second network has a recurrent convolutional network with the same convolutional, fully connected layers and a LSTM layer whit activation thanh. Furthermore, the hyperparameters were tuned in order to obtain the best results in the purposed Atari games. The results obtained after 10M trained episodes shows that this architecture obtain better results than others in specific environments (Atari games) [30].

Rezende et al. [31] treat the difficulties faced in accurately detecting cervical cancer through the conventional Pap smear test. Furthermore, the authors mentioned the importance of computational tools to support screening efficiently, especially during the current health crisis. The article said that machine learning has the ability to reduce the test limitations, but the lack of high-quality datasets has break the development of strategies to improve cervical cancer screening. The Center for Recognition and Inspection of Cells (CRIC) platform has created the CRIC Cervix collection, which currently contains 400 images of conventional Pap smears with manual classification of 11,534 cells. The dataset is a good beginning for improving machine learning algorithms to automate tasks in cytopathological analysis.

Mohammed et al. [32] propose the automatic analysis of pap smear test in order to hep underdeveloped nations with their lack of qualified pathologists. The aim of the project is to use pre-trained CNN image classifiers to categorize a single-cell in a pap smear image. The pre-trained DCNN is DenseNet169 which is a top-1 accuracy Keras Application. Furthermore, the project apply the three single-cell CPS image analysis piplines in order to classify the images or detect lesions in cells. The first pipline use

an additional segmentation stage than the second pipline in the classification using hand-crafted features The third pipline uses a deep learning approach for classification. Finally, after 100 epochs of training, using cross-entropy loss function, the same hyper-parameters for all the experiments, the DenseNet169 network obtain 0.990 of average accuracy, 0.971 of precision, 0.974 of recall and 0.974 of F1-score. The results were compared with another implementation, which use the same dataset and VGG19 network as feature extractor, obtaining ow results than the proposed paper.

Uzkent et al. [33] propose a reinforcement learning agent that chooses the spatial resolution of each image delivered to the detector in order to decrease the significant computational and financial costs associated with employing high spatial resolution images. Furthermore, they mention that the agent was trained in a dual reward setting to select low spatial resolution images for processing with a coarse CNN and when the image has high spatial resolution is processing with a fine CNN. The experiments were carried out with the xView dataset, which consists of huge images, in which the proposed agent improve the runtime efficiency by 50%, while using images with high resolution only 30% of the time, while keeping same accuracy as a detector that solely utilizes high resolution images. Finally, it concludes that using a reinforcement learning agent the dependency on images with high resolution can be reduced.

Alsalatie et al. [34] focus on cervical cancer screening and the challenges that traditional screening methods approaches face. Alsalatie mention the importance of computer-assisted diagnosis to improve the accuracy of cervical cancer screening. According to the paper, conventional screening methods take into account the knowledge of a pathologist which can result in misdiagnosis and low diagnostic effectiveness. In addition, the article propose a deep learning model designed for the automatic diagnosis of whole-slide images (WSI) in cervical cancer samples. The proposed network has a high accuracy rate of up to 99.6%, and consider the entire staining slice image, not just a single cell. The deep learning architecture considers overlapping and non-overlapping cervical cells in the WSI. Finally, they mention that the work is distinct from existing research in terms of simplicity, accuracy, and speed.

# Chapter 4

# Methodology

## 4.1 Data Description

In this section, it is presented information about the two datasets used in the project.

The first dataset contains 2300 images and it was retrieved from the Mendeley data repository where the samples were collected from 460 individuals from three Indian institutions in 2019.

- Dr. B. Borooah Cancer Research Institute
- Babina Diagnostic Pvt. Ltd
- Gauhati Medical College and Hospital

The tests were generated using liquid-based cytology technique rather from the traditional approach because it is needed high-quality images. Also, it is critical to get samples that are free of blood, mucus, inflammatory exudate, and lubricant in order to generate satisfactory ThinPrep Pap test specimen [35]. Thus, the test images were shot with a Leica DM 750 microscope that was linked to a customized ICC50 HD camera and its approved computer softwared. In addition, the microscope was adjusted at 400x magnification. As result, the dataset contains JPG high definition images with dimensions of $2048 \times 1536$ pixels [36].

With the first dataset, *Toapanta* built a new dataset by cropping the images into $150 \times 150$ pixels images, see Fig. 4.2. The dataset collect 100 images that contain cells and, conversely, images where there are no cells and only show the background of the test [37].

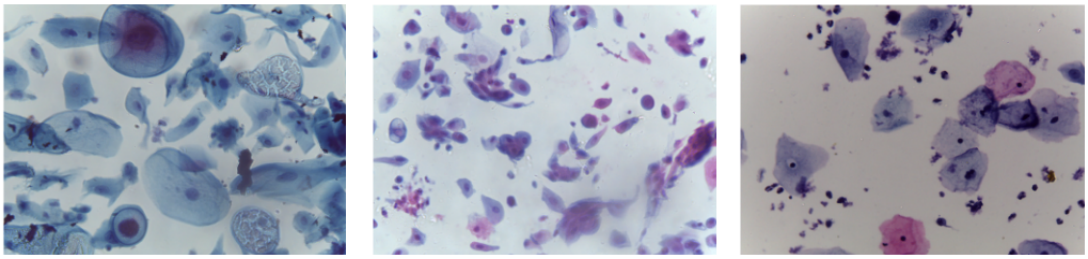
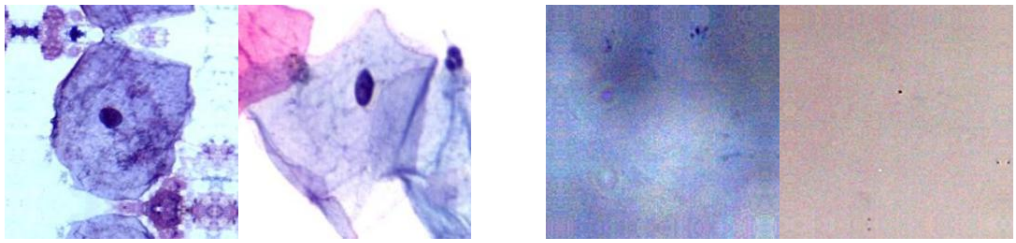


Figure 4.1: Three samples of the dataset retrieved from Mendeley data repository



Cells No cells

Figure 4.2: The right hand samples shows two cropped images with cells. The left hand samples shows two images of the background of tests

## 4.2 Cell Recognition Model

It is necessary to implement a CNN for the recognition of cells in a Pap smear test since this will help to define the reward signal in the environment. Therefore, it is required to generate the training, validation and testing data using the images from the first data set, see Fig. 4.2, as well as the CNN architecture.

### 4.2.1 Data Preparation

Currently it is known that for the training of a CNN, the data must be split into three different folders such as train with 80% of data, test and validation with the rest of the data. This is because, it is important prevent overfitting in the CNN since it is undesirable behavior that happens when a machine learning model predicts well for training data but not for new data [38]. Furthermore, the data was previously augmented because it was required to have wide diversity of data.

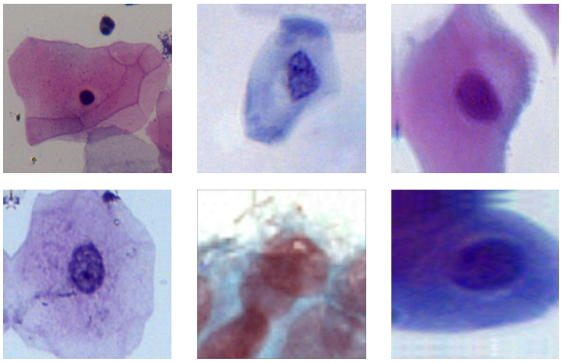


Figure 4.3: Set of images with cells used for training the cell recognition model

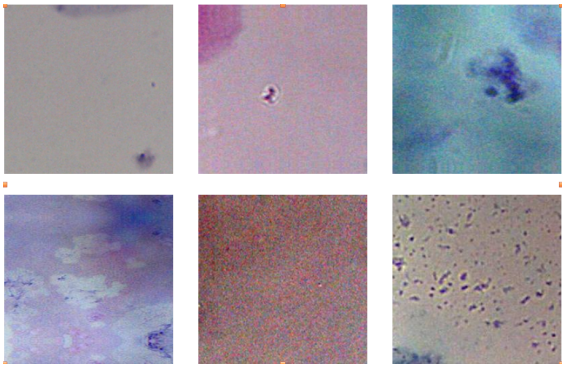


Figure 4.4: Set of images of background without cells from samples.

### 4.2.2 Convolutional Neuronal Network Architecture

As previously mentioned, this CNN will help us to define the reward signal in the reinforcement learning environment. Because of this, this convolutional network that is going to determine and classify the images positive for cell or non-cell, must be as fast as possible so as not to delay the training of the agent. Taking this into account, a most basic but efficient architecture was proposed for the cell detection task. The input layer is a convolutional layer with 10 filters of $2 \times 2$ which receives a $50 \times 50$ pixels image. The second convolution layer contains 32 filters of $2 \times 2$. The third convolution layer has 32 filters of $2 \times 2$. The last convolution layer has 64 filters of $2 \times 2$. Furthermore, between each convolution layer there is a max-pooling layer of $2 \times 2$. In addition, the last convolution layer is linked to a flatten layer followed by a dense layer of 16 neurons with a dropout of 0.2 and this in turn is connected to a dense output layer of 2 neurons and softmax activation function. All the other layers have the ReLu activation function, see Fig. 4.5.

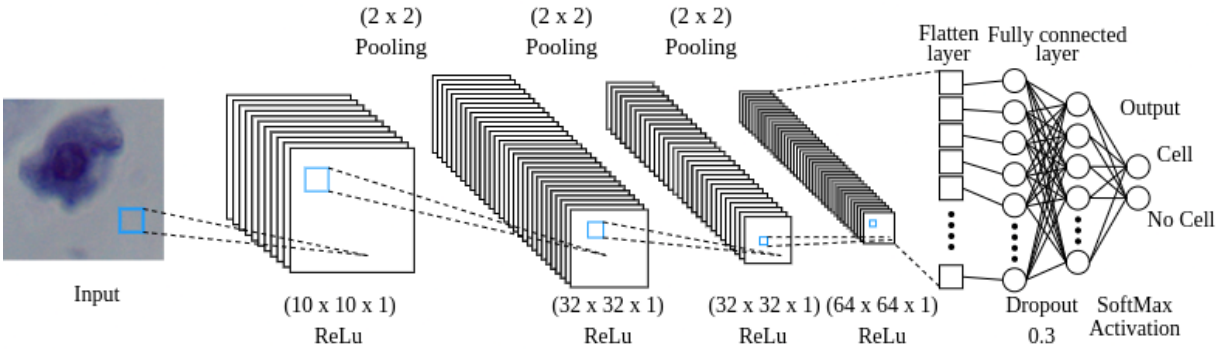



Figure 4.5: Graphic representation of the data flow through the CNN architecture.

### 4.2.3 Training

Before training, the images are rescaled into $[0, 1]$ interval and resized to $50 \times 50$ pixels in order to reduce computational resource and execution time. The images are then turned into tensors in accordance with TensorFlow specifications and grouped in batches of 32. Finally, the training is carrying out with 50 epochs because it is enough for achieving an accuracy over 90%.

## 4.3 Environment and Agent Design

This section focuses on explain how the agent and environment are built. The agent's purpose is to detect cells and collect the largest number of cells by moving around in the environment which are images of Pap smear tests.

### 4.3.1 Agent

In reinforcement learning there are many algorithms which can be used for training agents, it is important to choose the correct one to obtain good results during the interaction between agent-environment. The agent, in this graduation project, uses PPO because of this algorithm has demonstrated to be robust and perform well on a wide range of applications [24, 39, 40]. PPO is robust because it works well on a wide variety of tasks without depending on hyperparameter tuning, or the search for ideal hyperparameters [24]. In chapter 2 the PPO algorithm is covered in detail.

The PPO uses the natural CNN [41] that is designed with three convolutional layers, which are know as feature extractor, and two fully connected layers which maps the fea-

tures to actions/value. This network architecture is relatively small, and the decision to implement a small CNN was chosen since networks size influences in training times. Furthermore, the network architecture was judged appropriate since other image-based agents employ networks of comparable size and design [41, 42].

| **Layer** | **Conv. 1** | **Conv. 2** | **Conv. 3** |
|---|---|---|---|
| Input dimension | $360 \times 1080 \times 1$ | $89 \times 269 \times 32$ | $43 \times 133 \times 64$ |
| Kernel | $8 \times 8$ | $4 \times 4$ | $3 \times 3$ |
| Stride | 4 | 2 | 1 |
| Output dimension | $89 \times 269 \times 32$ | $43 \times 133 \times 64$ | $41 \times 131 \times 64$ |
| Activation function | ReLU | ReLU | ReLU |

Table 4.1: Summary table of the feature extractor in the agent's CNN architecture.

| **Layer** | **Fully connected** | **Value head** | **Policy head** |
|---|---|---|---|
| Input dimension | 343744 | 512 | 512 |
| Output dimension | 512 | 1 | 4 |
| Activation function | ReLU | Linear | Linear |

Table 4.2: Summary table of the dense layers in the agent's CNN architecture.

In case of PPO algorithm, it uses a shared network because the value function and the policy are approximated. This indicates that both approximations use the same base network. Tables 4.1 and 4.2 gives more details about the CNN described before.

### 4.3.2 Environment

The environment is the collection of 300 Pap smear test images. The images are read and prepossessed using Open Source Computer Vision Library (OpenCV) which is a library specialized in computer vision and machine learning [43]. The images are resized to $1080 \times 1080$ pixels in order to standardize the environment. Each image contain since 4 cells until more than 20 cells in there. The agent is a ROI (region of interest) which can interact with the environment by moving in it and capturing cells. The agent can perform 4 actions which are described in the Table 4.3. The four actions are discrete and moves the ROI 25 pixels each action. Furthermore, the environment shows 4 windows, the "Test" window is where the agent is moving around searching for cells, in addition, this window has linked

a "ROI" window which shows in more detail where the agent is located. The window "Detection" shows where the agent has detected a cell and it is boundary by a square, in the same way this window has a "ROI" window where the detected cell is showed in more detail, see Fig. 4.6.

| Action Nr. | Action |
|---|---|
| 1 | Right |
| 2 | Left |
| 3 | Up |
| 4 | Down |

Table 4.3: Agent's actions.

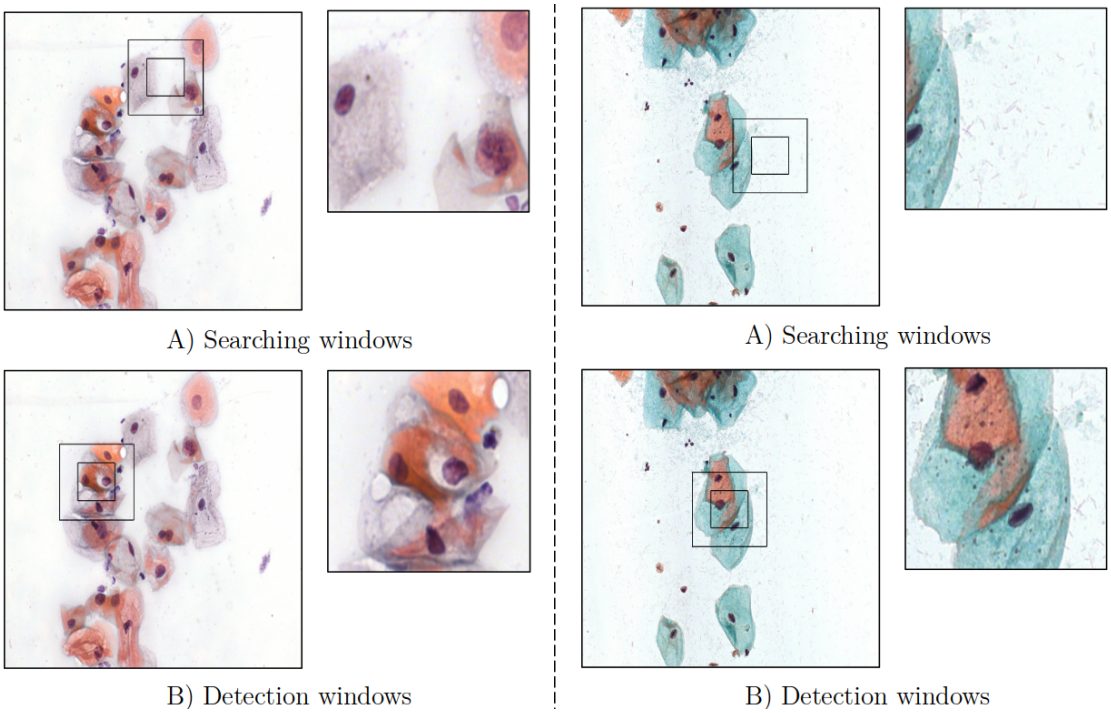



Figure 4.6: Graphic representation of two environments. The environment has two different parts, the A part shows the searching windows where the agent is moving and what the ROI capture is shown in the right window. On the other hand, The B part shows the detected cells bounded by a black square, and the detected cell is represented, in more detail, in the left window.

The observation space given to the agent are three stacked gray scale images of the test. Each stacked image shows one different step of the ROI what helps to the PPO algorithm to learn better, see Fig. 4.7. Therefore, the shape of the observation space is $1080 \times 360 \times 1$ where the range of the values is $0 - 255$.

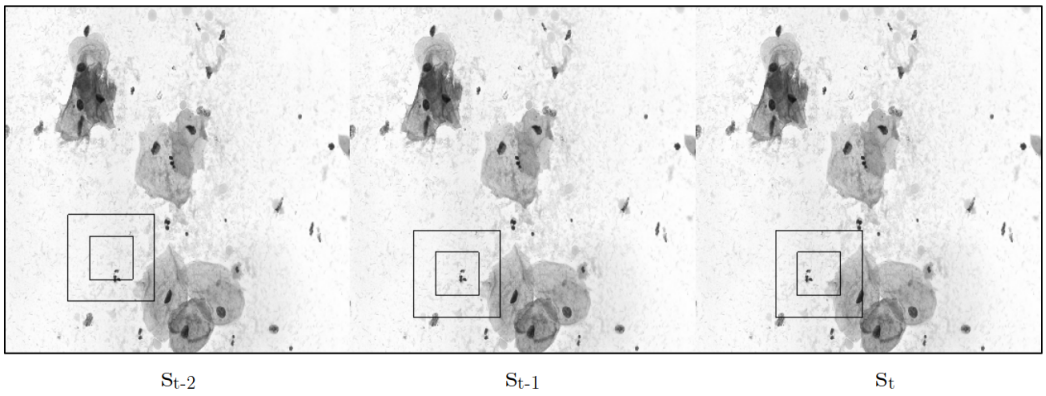



Figure 4.7: Graphic representation of the observation space. The right hand side figure shows the current state $s_t$, while the middle. $s_{t-1}$ and the left hand side, $s_{t-2}$, figures shows the previous states

### 4.3.3 Resetting the Environment

An episode conclude when the agent reaches the 1024 time steps. When this occurs the environment is reset. This indicates that the ROI is positioned at the starting point and the reward is restarted. Furthermore, a new Pap smear test image is selected from the dataset and the episode start again.

### 4.3.4 Goal and Reward Signal

The agent's purpose is to explore its environment for a target. When the agent find a cell this is marked by a boundary square and the agent keeps searching for another cell and so on. The reward increase according to how many cells the agent has founded. While the agent does not find a cell it receive a penalty, this force the agent to search and explore in the environment. Furthermore, staying in one cell will also result in a penalty, preventing the agent from being stuck in a single cell and promoting the detection of the greatest number of cells in the samples, see the Fig. 4.9. In addition, the agent recognizes the cells thanks to the CNN trained for cell detection, see chapter 4.2 and Fig. 4.8 for more details.

This kind of reward signal is known as discrete reward. A discrete reward signal alters in a discontinuous manner in response to changes in the environment, observation, or actions. This type of reward function usually has a slow down convergence and need more sophisticated network architectures. Discrete rewards are often implemented as environmental events, for instance when an agent perform good actions it receives a positive

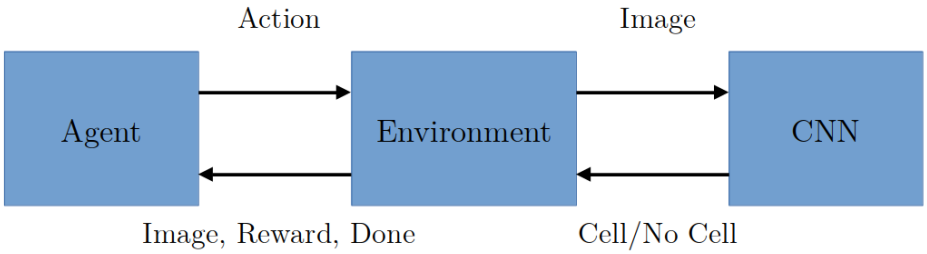
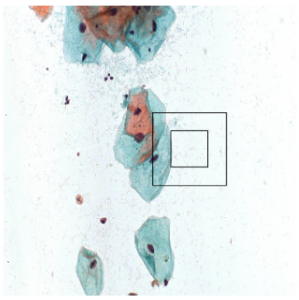



Figure 4.8: Interaction between the agent, cell recognition CNN and the environment

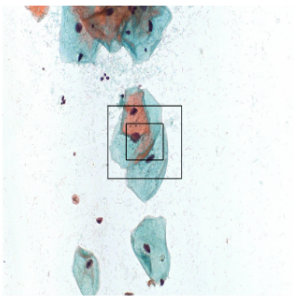
reward, on the other hand the agent obtains a penalty when perform bad actions [44]. The reward function for this environment is represented as follow

$$R = r_1 + r_2 + r_3 = \begin{cases} r_1 = -0.5 \quad , & \text{if agent is searching for cells} \\ r_2 = 10 \times n_{cells} \quad , & \text{if agent detect a cell} \\ r_3 = -5 \quad , & \text{if agent detect the same cell multiple times} \end{cases} \tag{4.1}$$

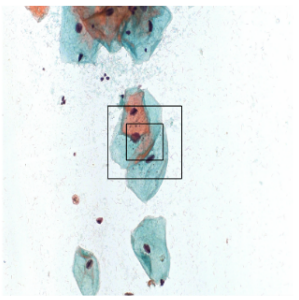
where $n_{cells}$ is the number of detected cells.



a) Penalty due to ROI is not in the cell.



a) Reward due to ROI found a cell.



a) Penalty due to ROI is detecting the same cell.

Figure 4.9: Different situations when the agent receive a reward or penalty.

## 4.4 Implementation

In this section it described the reinforcement learning model for cell detection. Typically, in a digital Pap smear there are around 5 to more than 20 cells in there. In that case, where there are many objects that should be detected, it is not efficient use a CNN for detecting all that amount of cells. Taking into account this, the project proposes a RL agent which locate as many cells as possible without using external matrices for storage the detected cells [37].

### 4.4.1 Software

Python was chosen as the programming language since it is flexible and has a good variety of scientific packages for doing research [45]. Besides, python is a interpreted programming language that allows several computing programming models such as object oriented programming [46], which was used for programming the environment following the API of Gym [27]. Further, the agent was trained using the PPO algorithm from Stable Baselines 3 [47]. The cell recognition model was designed and trained using TensorFlow which is becoming one of the most popular deep learning frameworks due to all of the tools and pre-built functions that make neuronal development easier. Besides, TensorFlow smooth the way for training due to the facility of setting hyperparameters and monitoring results of the machine learning models [48].

### 4.4.2 Hardware

It is necessary to mention that for this graduation project was used a NVIDIA A100 SXM4 40gb because of the computational time and complexity for training agents is high.

### 4.4.3 Pseudocode

At the start of a training session, the agent's weights are initialized and the environment is reset. The agent then obtains the first observation. As the agent steps through the environment (known as a trajectory), it collects samples. This process involves the agent determining an action based on the current observation, and the environment interacting with the graphics engine to produce a new image and calculate the reward. The new image

then becomes the next observation. After collecting a trajectory, the agent's weights are updated using the PPO algorithm. The process of stepping through the environment continues until the episode ends. Then, the environment resets and the training starts again.

**Algorithm 3:** Training flow

```
Initialize agent and its weights;
while train do
    Reset environment and gather initial observation S;
    while episode not done do
        for time step t = 0, 1, ··· , T do
            Let agent choose action A based on state S;
            Update graphics engine according to action A;
            Get new image (State S′) from graphics engine;
            Calculate reward R;
            Calculate advantage Â;
            Check if episode done;
            S ⟵ S′;
        end
        Update weights with PPO;
    end
end
```

## 4.5 Performance

The effectiveness of an agent is determined by its ability to perform or complete the designated task. An precise performance metric is difficult to establish, but one approach to indicate an agent's performance is the reward it receives. Because the agent attempts to maximize the reward, a large reward would, in most situations, indicate that the agent is working well. Moreover, while evaluating an agent's performance, the incentive was utilized to determine how well it performed. However, the compensation is unlikely to be sufficient to decide an agent's success on its own. As a result, in addition to the incentive, a visual assessment was undertaken by personally monitoring the agent in action to assist identify the agent's behavior.

## 4.6 Experiments

The experiments cover three stages, see Fig 4.10. The fist stage follows the next parameters and training process.

- Eight agents will be trained, and the hyperparameters used for the first four agents (A-D) are in the Table 4.4 with $n\_steps = 512$. On the other hand, the other four agents (E - H) use the same hyperparameters in the Table 4.4 with $n\_steps = 1024$. The agents A and E have not any change in the reward signal 4.1 . The agents B and F do not receive any penalty when detect the same cell more than one time. The agents C and G has not penalty while are searching a cell. The agents D and H have a high penalty while are searching for a cell. In order to compare the learning process and behaviors, all agents were trained 3 millions time steps, 4.5.

| **Hyperparameter** | **Value** | **Description** |
|---|---|---|
| learning_rate ($\alpha$) | 0.0003 | Progress remaining, which ranges from 1 to 0. |
| n_steps | 512; 1024 | Number of steps per update for each environment. (trajectory) |
| batch_size | 128 | Number of images processed by the network at once |
| n_epochs | 10 | Number of updates for the policy using the same trajectory |
| gamma ($\gamma$) | 0.99 | Discount factor |
| gae_lambda ($\lambda$) | 0.95 | Bias vs. variance trade-off |
| clip_range ($\epsilon$) | 0.2 | Range of clipping |
| vf_coef ($c_1$) | 0.5 | Value function coefficient |
| ent_coef ($c_2$) | 0.0 | Entropy coefficient |
| max_grad_norm | 0.5 | Clips gradient if it gets too large |

Table 4.4: The hyperparameters included in the PPO Stable Baselines 3 implementation, with descriptions and values. The symbol following a hyperparameter name refers to the coefficient.

- The second stage will be to retrain the agents for 2 millions more steps which obtain highest results and which shows good behavior in the manually testing. The same hyperparameters that were mentioned in the previous stage will be used for each agent. This stage help us to reject the incorrect defined reward signals while we retrain the agents with high potential to learn the task.

| Agents | Description |
|---|---|
| **A , E** | Using the reward signal without changes. |
| **B , F** | No penalty for detecting the same cell multiple times |
| **C , G** | No penalty while searching for a cell |
| **D , H** | High penalty while searching for a cell, $r_1 = -10$ |

Table 4.5: Brief description of the agents.

- Finally, the third stage will be to retrain for 6 millions more time steps the agent which shows the best results from the previous stage. The same agent will be retrained 4 times in parallel, obtaining 4 more agents from the initial model.

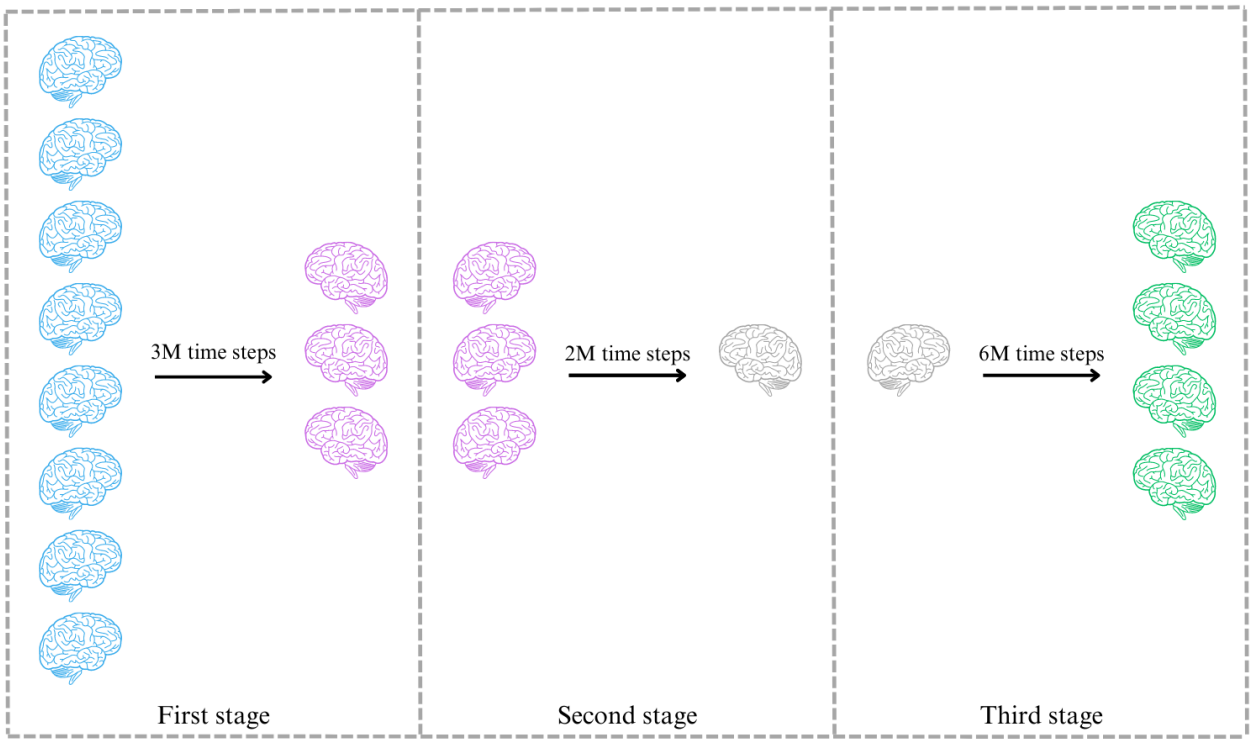



Figure 4.10: Graphic representation of the experiments.

# Chapter 5

# Results and Discussion

In this chapter, the outcomes of training and assessing the agents discussed in Chapter 4 are presented. The initial part shows the training outcomes using graphs that demonstrate the alteration of mean reward for each agent during training. Following that, the conduct of the agents is explained. Since the policy or behavior of the agent is challenging to illustrate through data and graphs, this section provides descriptions of their behavior, based on personal observation of the agent in operation. All the agents were trained for 3 million steps and the training duration of approximately 24 hours for every 1 million time steps trained.

## 5.1 Training of the Agents

### 5.1.1 Firs Stage

The performance of the eight agents during training is shown in Figures 5.1 and 5.2. The plots illustrate changes in mean reward per two episodes over time, with the $x$-axis representing time steps and the $y$-axis representing the mean reward. The reward received by the agents is influenced by the randomized environments, which means that the data shows significant fluctuations depending on how many cells contain the test which is acting as environment in that episode. In addition, in order to compare the training process between agents, the performance of the first set of agents is shown in a single figure 5.3 and the performance of the second set of agents is shown in another figure 5.4.

Analyzing the figures 5.1 and 5.2, we can see that all the agents start with a high

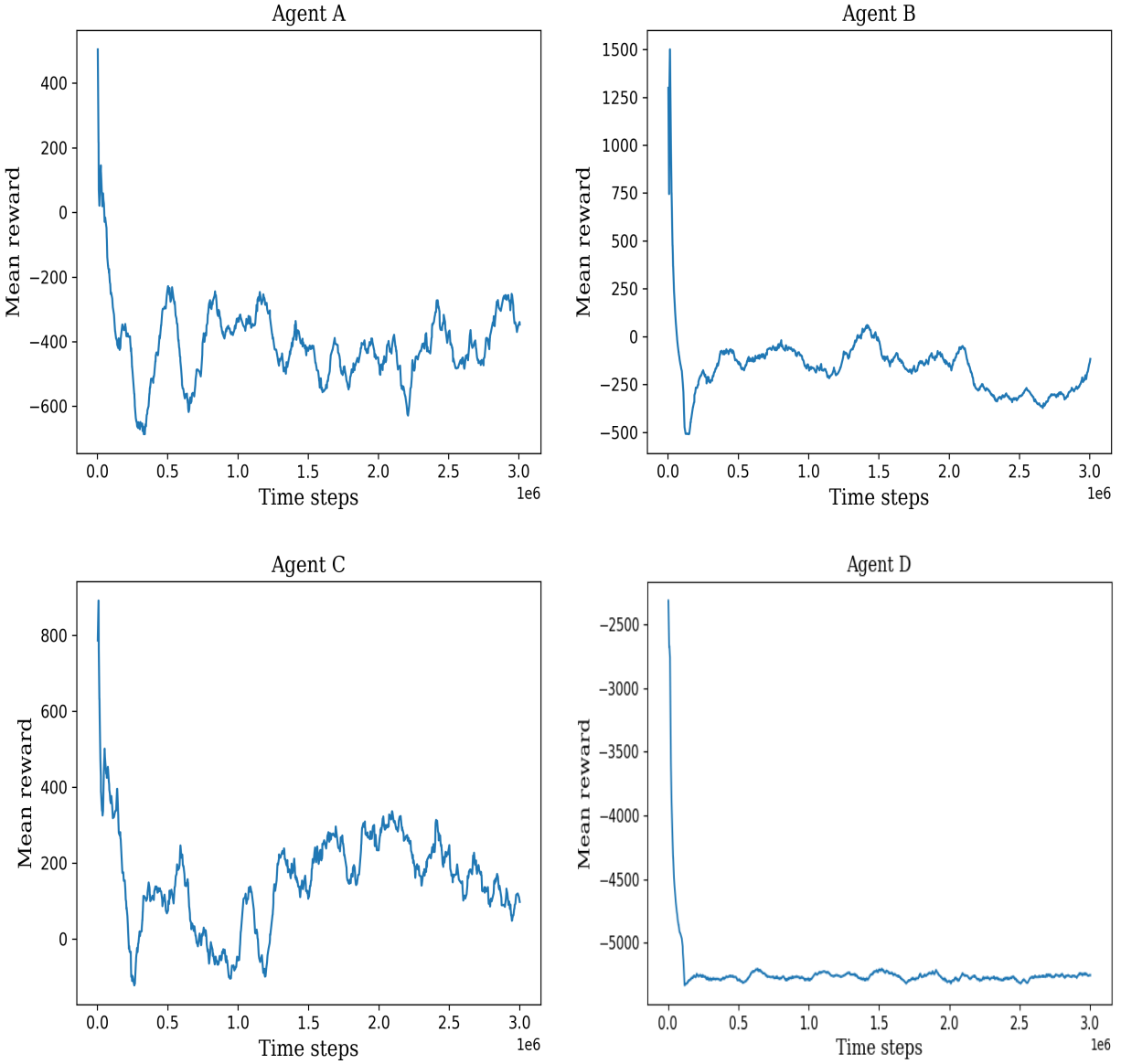



Figure 5.1: The figure shows the first four agents using the hyperparameter n_steps equal to 512. The plots represent the obtained score at the end of each episode. The $x$-axis shows the time steps, until 3 million steps. The $y$-axis shows the score obtained in the episodes.

mean reward and then it drops rapidly. This behavior is caused because the policy at the beginning is more "random" and cover more parts in the test. So, from the first set of agents, we can mention that the agent C has positive rewards after decay and shows a stability in 150 mean reward mark. In the second set of agents, the agents E, F and G their mean reward fluctuates between +500 and -500. The agents D and H have an abrupt comedown and they are stable in -5200 mean reward mark.

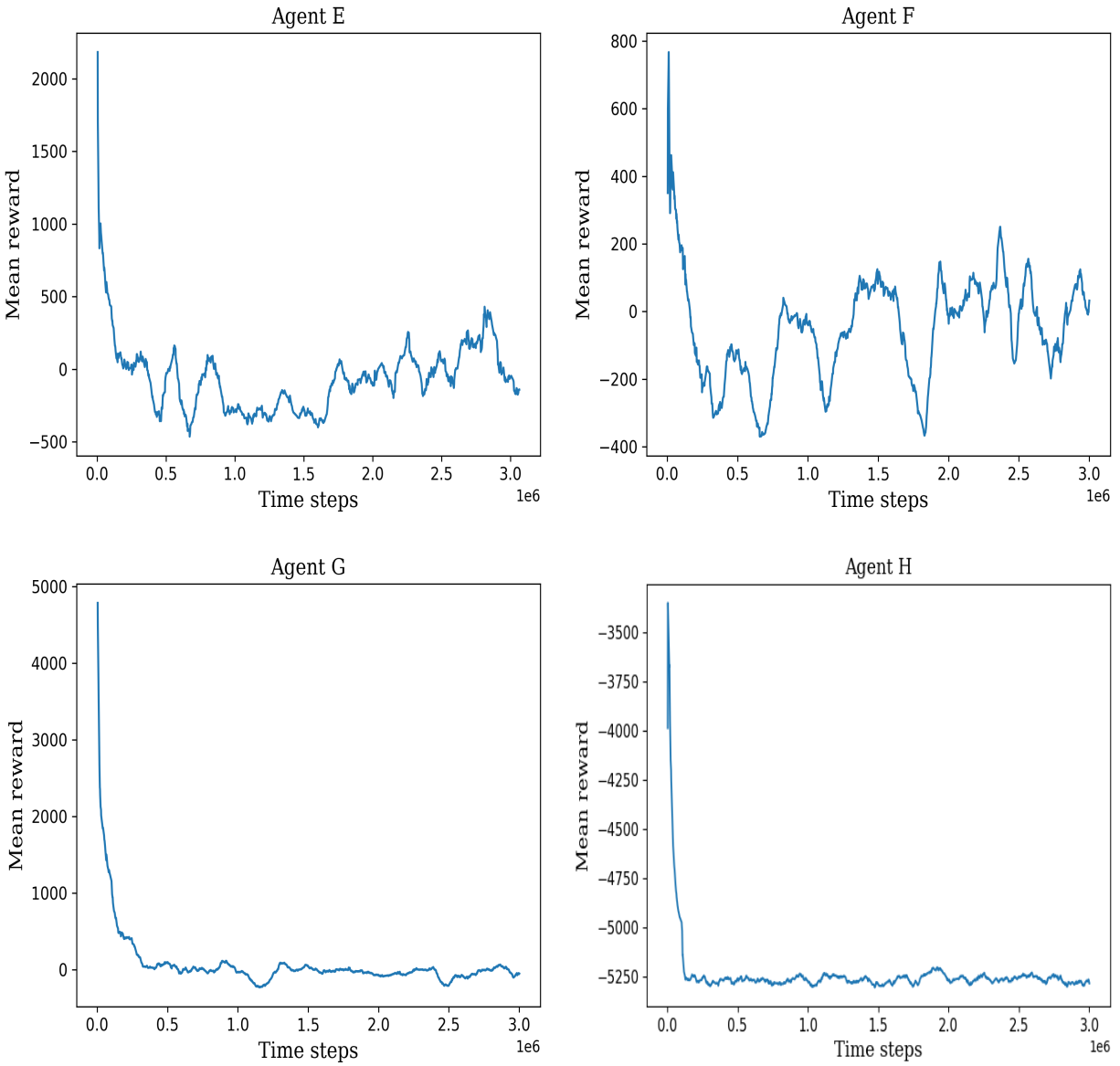



Figure 5.2: The figure shows the second set of agents which use the hyperparameter n_steps equal to 1024. The plots represent the obtained score at the end of each episode. The $x$-axis shows the time steps, until 3 million steps. The $y$-axis shows the score obtained in the episodes.

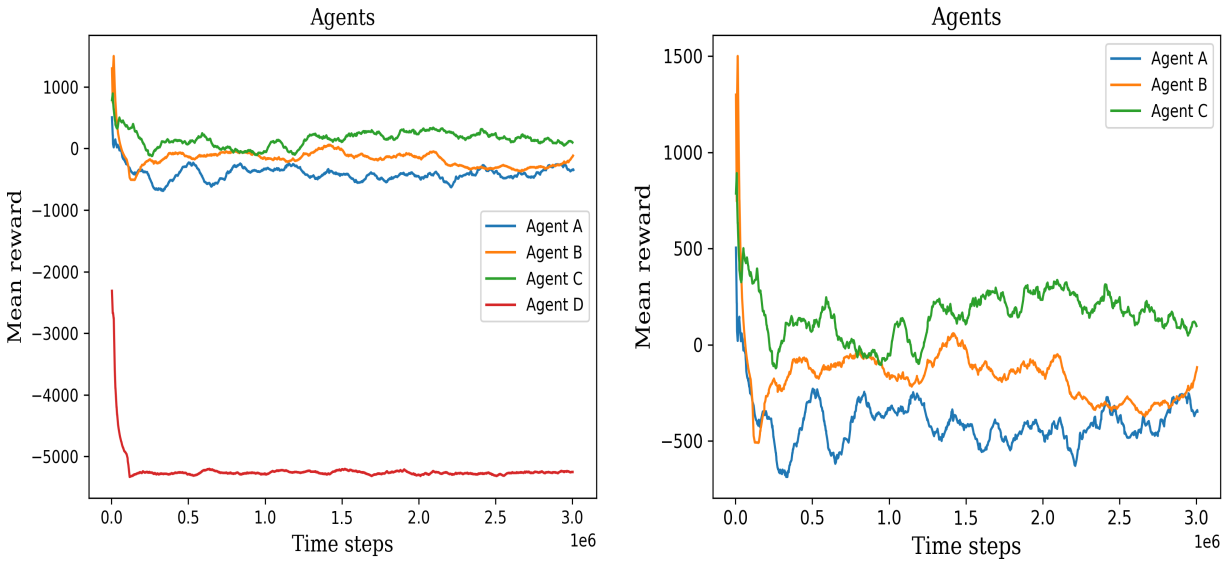



Figure 5.3: The right hand side plot shows all the first set of agents scores, permitting the ability to visually compare better their training process. The left hand side plot shows only three agents, the agent D was not considered because it has lowest values.

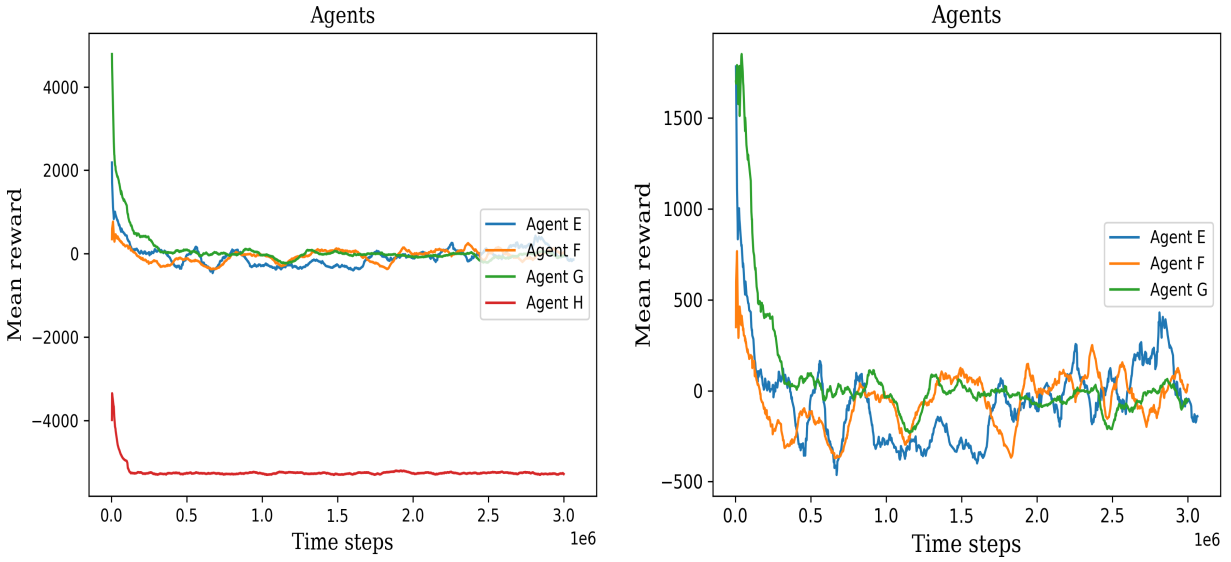



Figure 5.4: The right hand side plot shows all the second set of agents scores, permitting the ability to visually compare better their training process. The left hand side plot shows only three agents, the agent H was not considered because it has lowest values.

### 5.1.2 Second Stage

For this stage were chose three agents, which presented higher mean rewards during 3 million time steps, from the previous stage. The agents C and E experimented a mean reward decline when the retraining process start, this behavior is explained because of the environments vary the number of cells and the position of them. After that obstacle the

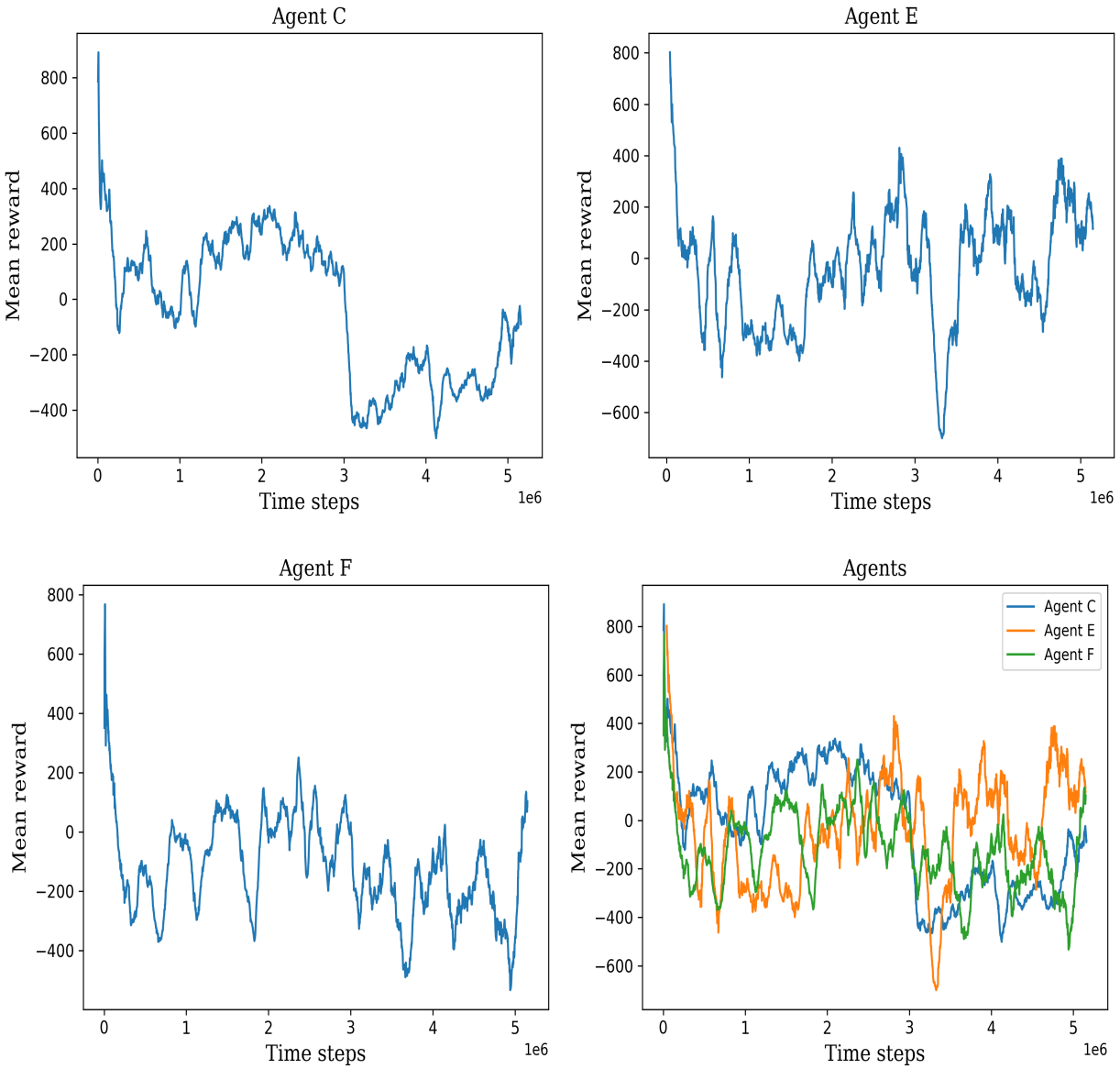



Figure 5.5: The graph shows the retraining results of the agents C, E and F which showed good results and searching behaviors. The left bottom figure shows the merged results of the agents for a better visualization of the training process.

agent E could recover high mean reward values during the rest of the training. On the

other hand, the agent C could not recover the previous behavior causing low mean rewards values. Furthermore, the agent F has a behavior which does not present a growth in the mean reward values obtained, in a range of -400 and 200 points. Finally, comparing the results of the retrained agents, see Fig 5.5, we can conclude that the agent E has potential to achieve an expected behavior.

### 5.1.3 Third Stage

In this stage the agent E was retrained four times in parallel for 6 million more time steps. In the Fig 5.6 we can see that even the hyperparameters and the pre-trained model is the same in the four cases, the learning process can vary. Furthermore, from the four agents we can chose one agent which achieve better results than the other agents in this case E1.

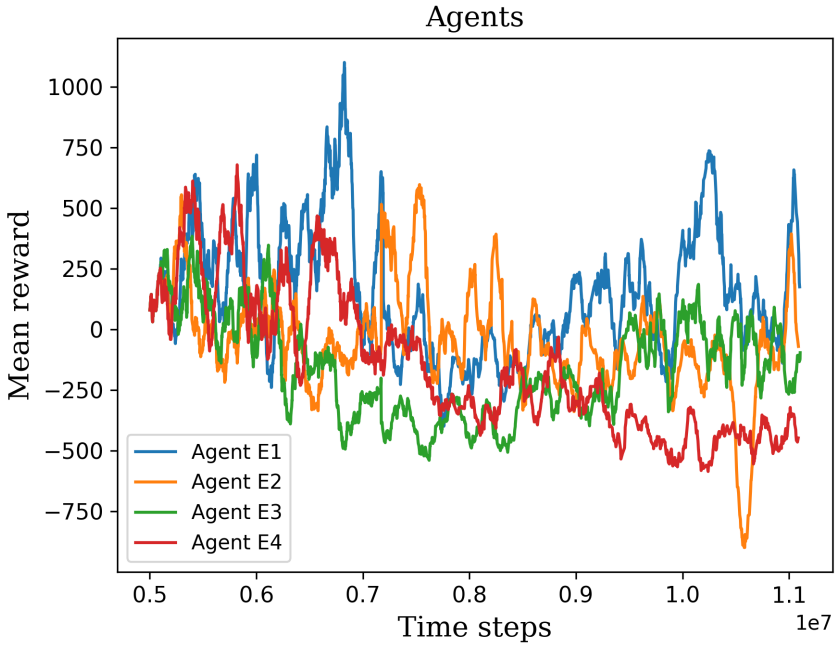



Figure 5.6: The graph shows the retraining results of the agents C, E and F which showed good results and searching behaviors in the previous stage.

## 5.2 Agents Behavior From Manually Testing

In this section, the trained agents' behavior and performance are analyzed and discussed. In order to get a better understanding of how the agents are learning all the agents were

tested in 100k, 500k, 1M, 2M and 3 millions time steps. But this section only consider the best results.

### 5.2.1 First Stage

By manually testing the agents we can conclude that in the first set of agents only the agent C has a smart behavior. The agent C has a fluid movement, and randomly get stuck in a repetitive behavior. Furthermore, the agent C shows that perform actions in order to get rewards detecting cells. The rest of agents in this first set of agents have the mentioned problem. The worst reward function for training can be seen in the agent D, which does not perform any action, agent D is only stuck in the starting point. The agent A has the second worst behavior because of only perform repetitive movements and get in a loop during the scanning process. Finally, the agent B when find a cell it get stuck in there, this was an expected behavior because of this agent has not setup a penalty when it get stuck in a specific cell.

In case of the second set of agents, the best build behavior was from the agent E which never get stuck in any part of the test and never perform loop actions. In addition, the agent E after finding a cell keep searching for more cells and perform actions in order to get the maximum number of cells. The agent G shows better results as well, but in some cases it get stuck in a part of the test or perform loop actions. Agent F has similar behaviors as agent B because of when it detect a cell it randomly search for another cell. Agent H has the worst behavior because the penalty while he is searching is to high what destroy the policy causing that the agent does not perform any action.

- First set of four agents.

  The agent A after wait a long time to start moving, then it start moving, firstly in the top of the environment, and when it reach the first cell it got stuck in a loop of movements around the cell. In some cases, the agent explore other areas in the test. Very rarely, the agent breaks the loop and perform different actions.

  The agent B start moving rapidly and has a searching behavior. Agent B perform up - down actions searching for a cell, then when it detect a cell it got stuck there doing the same actions, goes from the right to the left around the cell until the end of the

episode. This behavior is because the agent B does not have a penalty for detecting the same cell multiple times.

The agent C has the best behavior in the set of agents. Agent C start for performing down actions and then goes to the middle of the test and search for cells. This agent does not got stuck when detect a cell but when it touch the top of the environment it stand still there for a few steps. The penalty during searching was not defined, we can say that it encourage the agent to keep searching.

The agent D has the worst behavior in this set of agents. The penalty during searching was higher than the other agents what destroy the policy and it built an agent who only got stuck in the starting point along the entire episode.

These agents has an n_steps = 512 what means that the parameters where updated twice during an episode. This abrupt policy update cause that only the agent C could build a good behavior.

- Second set of four agents.

  The agent E start moving in the $x$-axis in the top of the environment, but then start to searching and he does not got stuck in one cell. Agent E in some cases repeat actions around a cell, after few steps he start searching again. In addition, the agent keeps searching even he touch a edge. Furthermore, it seems that the searching penalty encourage the agent to find rapidly a cell and the penalty for detecting the same cell cause that the agent does not get stuck in a particular area.

  The agent F start moving in the $y$-axis in the environment and then perform search actions in the middle of the environment. This agent got stuck in an area for many different environments, but other times he search for cells during the whole environment. It is because the penalty during searching is not apply. Generally, the agent has a searching behavior.

  Unexpectedly, the agent G since the episode start it perform loop action from right to left during the entire episode without signs of searching behavior. This behavior could be explained because the searching penalty was not defined for this agent.

  The agent H in the same way of D only stay still in the corner of the environment

without doing any action. Here we can claim that high penalties could burst in the process of learning.

These agents were trained using a n_steps=1024 what means that the policy is updated after one episode, letting time to the agents to discover more environment before update the policy.

- Remark

  If we compare trained agents, without counting D and H agents, with an untrained agent, see Fig 5.7, we can say that the trained agents show fluid movements and appear to acquired a searching behavior. We do not provide the graphic behavior of the eighth behaviors because of the number of time steps is not enough for visualize changes in the behavior of the agents. However, the results presented in the Figs 5.3 and 5.4 are important to see the learning process using different reward signals.

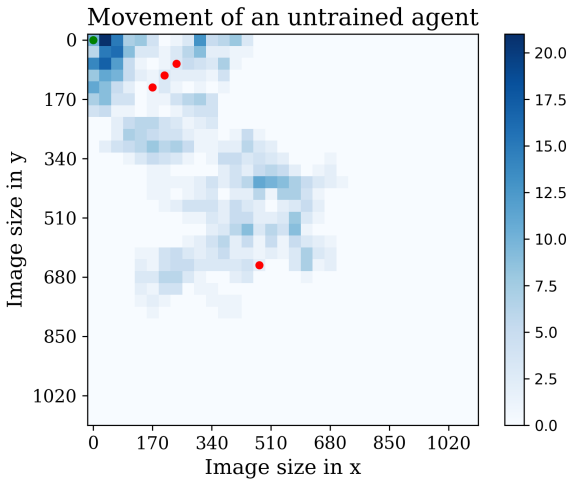
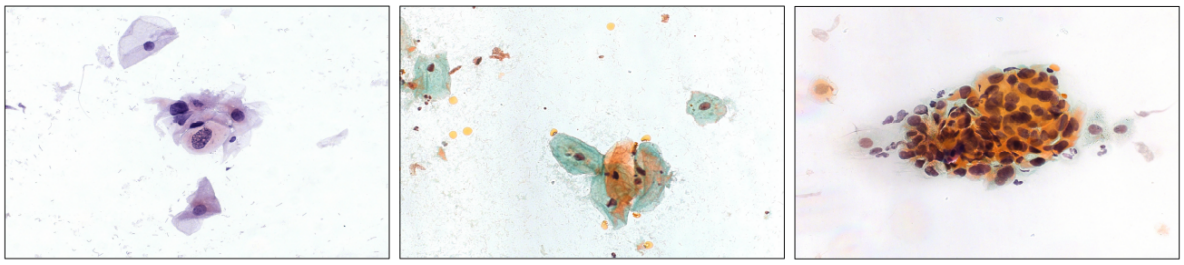



Figure 5.7: The figure shows how the untrained agent perform random actions and it has a frenetic behavior. The red dots are cells that the agent found during the episode. The blue bar indicates how many times the agent pass trough the same position.

### 5.2.2 Second Stage

For testing the agents we define three environments because doing this it can be tested the searching behavior learned for the agents, see Fig 5.8. The environments have cells located in different positions in the sample, causing that the agent perform actions for exploring and exploit the knowledge.



Environment 1 Environment 2 Environment 3

Figure 5.8: The figure shows three different environments which were used for testing the agents in the second stage and third stage.

- Agent C: As we can see in Fig 5.9 which illustrates that the agent C moves with fluidity towards a specific point, where it then enters a loop and repeatedly passes through that point over 250 times. However, in Environment 3, the agent demonstrates excellent behavior by successfully navigating through multiple cells before ultimately getting stuck in a loop. Furthermore, it appears that the agent has learned a pattern

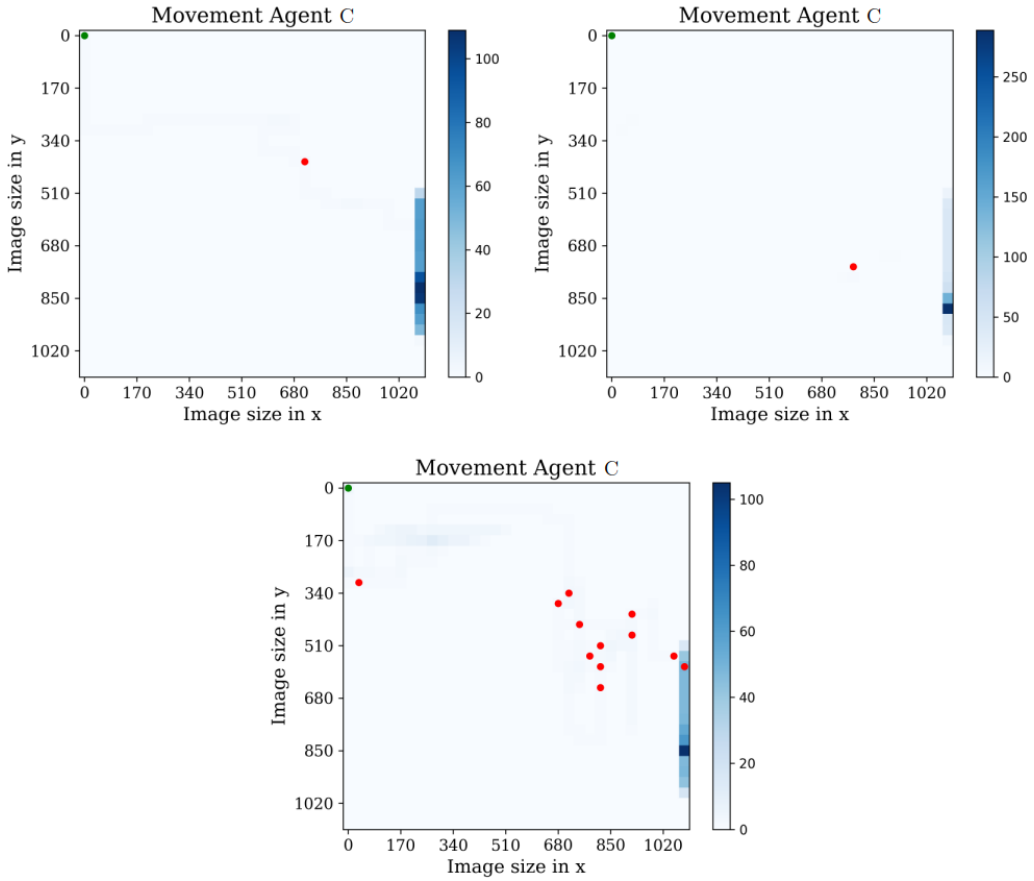



Figure 5.9: The figure shows the tracking of the agent C, while searching in the three mentioned environments.

for its search strategy, which involves first moving downwards and then towards the center of the environment.

- Agent E: Previously, the agent demonstrated a tendency to achieve high mean reward values, see Fig 5.5, which is reflected in its more thoughtful and fluid movement as seen in the figure 5.10. In the first two environments where Agent C got stuck, Agent E exhibited better exploration by covering more cells. However, in Environment 3, although it covered a significant number of cells, it got stuck in a loop and passed through the same point over 800 times. Finally, the agent's behavior is deemed satisfactory, and it appears to follow a search pattern of starting above the environment and then moving to explore the rest of the test.

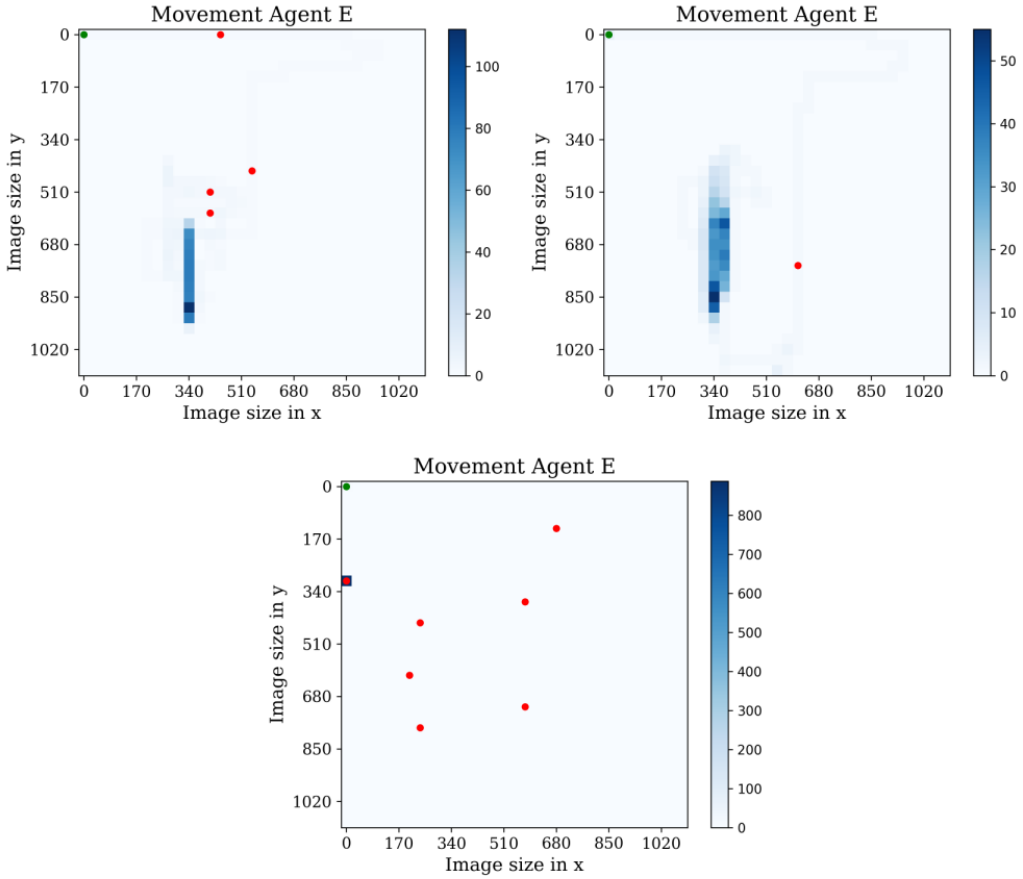



Figure 5.10: The figure shows the tracking of the agent E, while searching in the three mentioned environments.

- Agent F: After analyzing the training results, it became evident that the agent F did nos experience any significant increase in its mean reward values, see Fig 5.5.

Instead, it showed a significant variability, especially in environment 2, see Fig 5.11, behaving quite similarly to an untrained agent, see Fig 5.7. Even while it did not significantly improve in different contexts, it eventually became inert and repeated the same activities more than 700 times. Furthermore, any behavior of searching like showed the previous agents which has a searching pattern, did not develop. Based on these observations, it can be concluded that the additional 2 million time steps did not result in any noticeable improvement.

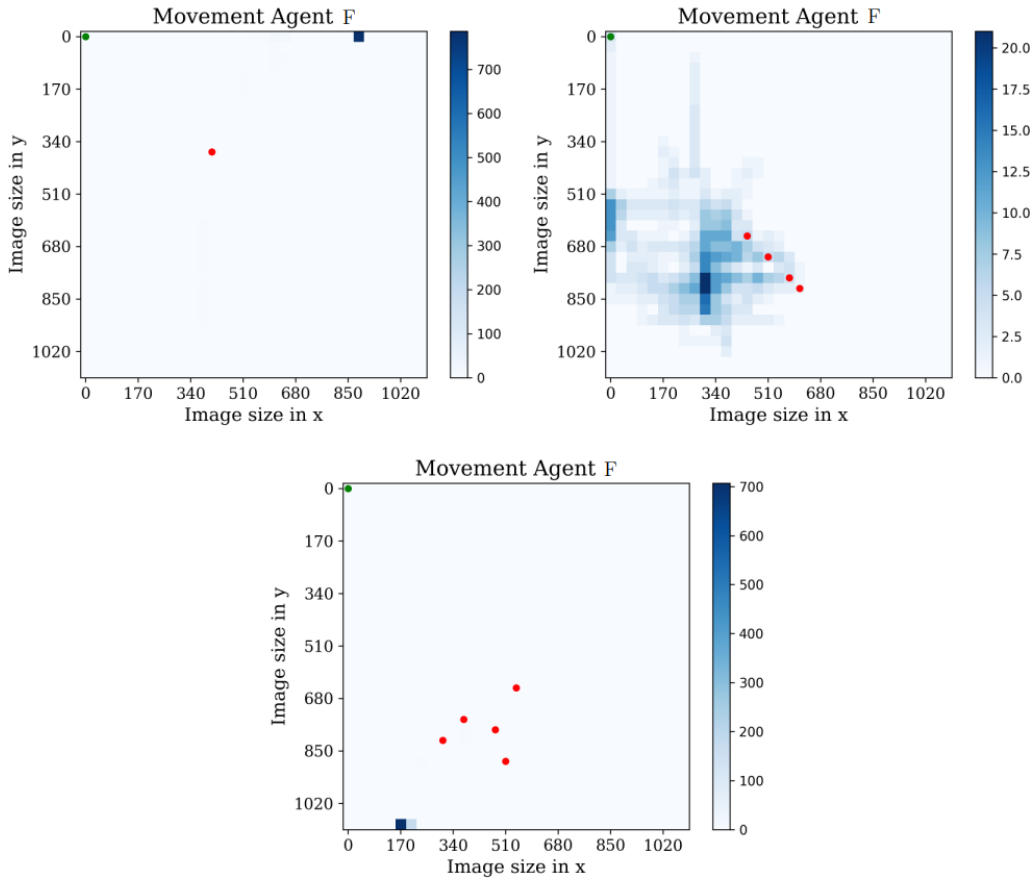



Figure 5.11: The figure shows the tracking of the agent F, while searching in the three mentioned environments.

### 5.2.3 Third Stage

In this last stage of training, agent E was chosen for retraining an additional 6 million time steps because it shows good results from the data and the manual testing. Furthermore, 4 agents with the same agent E model as seeds were retrained. Then, the four agents were tested and tracked in the defined environments, see Fig 5.8, and the outcomes are

presented below.

- Agent E1: While the E1 agent demonstrated outstanding results in the data achieving high mean reward values, see Fig 5.6, it did not perform any action in two of three environments and got stuck in the initial position, see Fig 5.12. However, in the third environment, the agent performed exceptionally well, finding 30 points where were detected cells. The agent E1 were tested several times in the first two environments, consistently producing the same outcome getting stuck in the started point. Notably, the agent E1 took strategic actions to locate the highest possible number of cells, going first down the environment to then start the search.

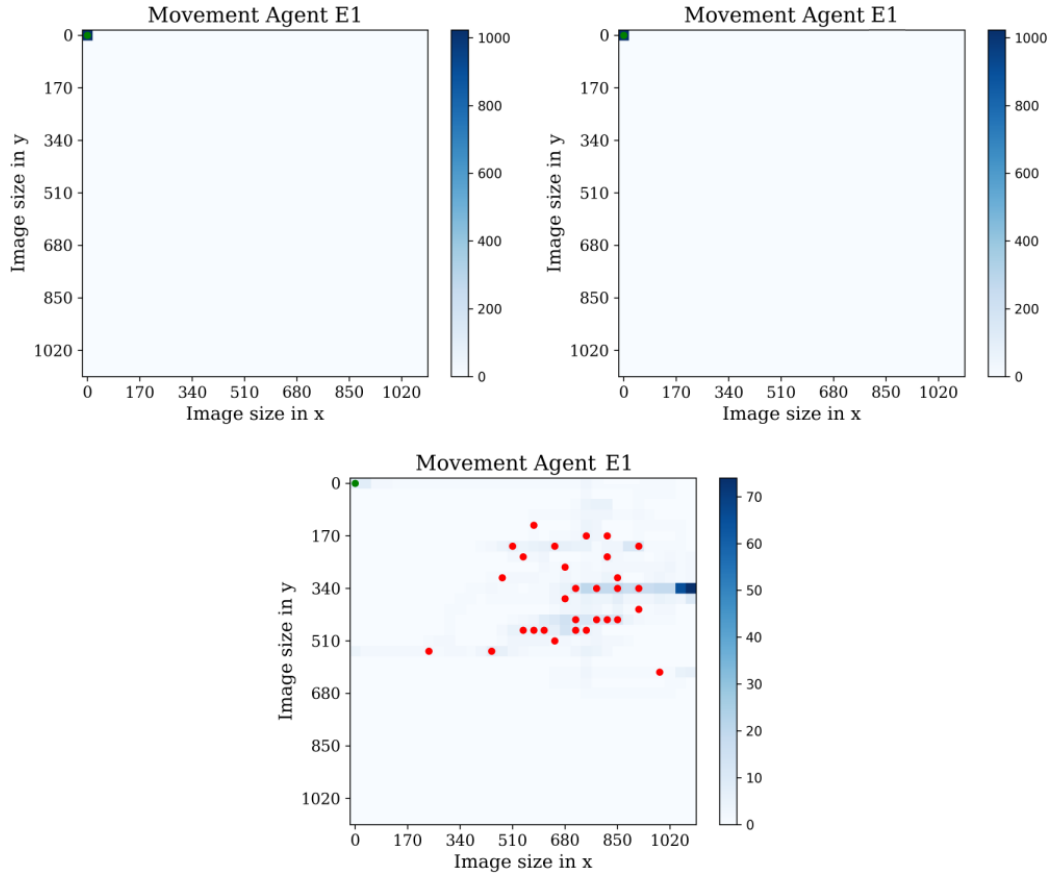



Figure 5.12: The figure shows the tracking of the agent E1, while searching in the three mentioned environments.

- Agent E2: In the same way as agent E1 the agent E2 shows high mean reward values. Testing the agent E2 in the environments it shows that in the first two environments he performs a very good search action, reaching 8 and 12 points where a cell is

found. Although, it is seen that in a specific point or actions, during the search, it got stuck in a loop, passing through the same point more than 250 times. However, in environment 3 the agent carried out an excellent search, finding more than 15 points for a very wide area of the test and the agent only pass 20 times through the same point, see Fig 5.13.

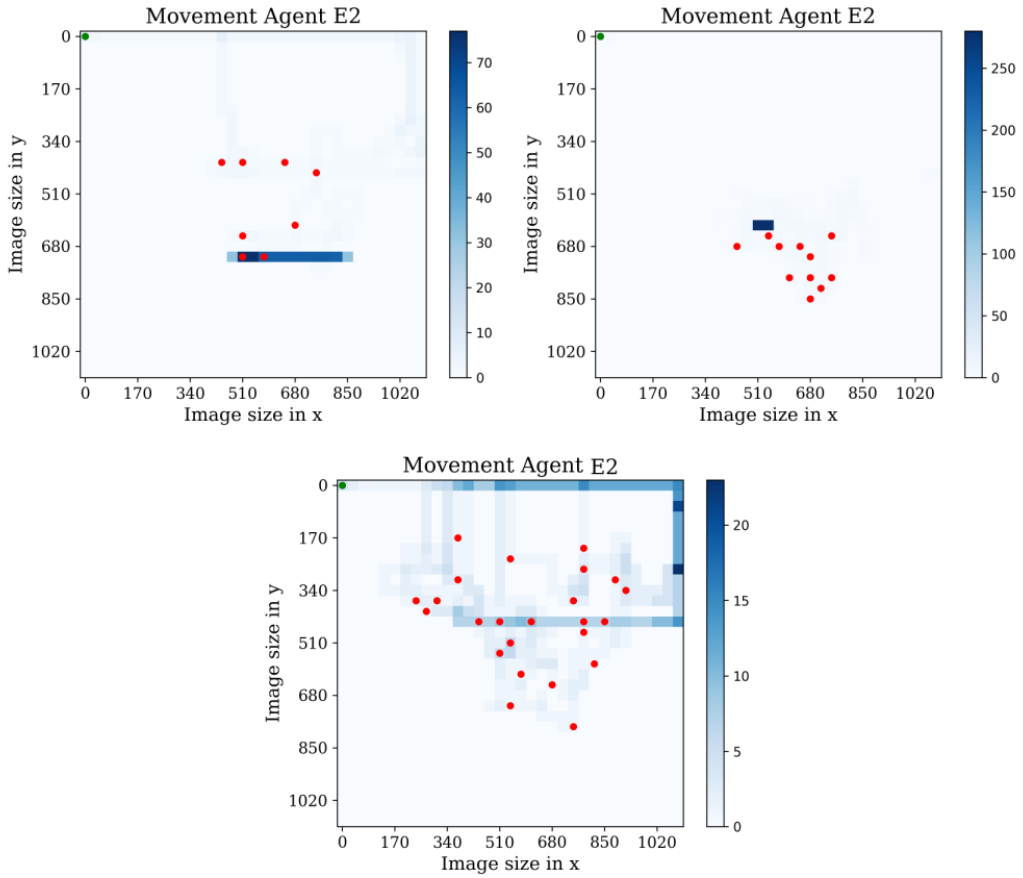



Figure 5.13: The figure shows the tracking of the agent E2, while searching in the three mentioned environments.

- Agent E3: The agent E3 shows a regular behavior in the training process, getting low mean reward values which are in range -500 and +500 points. After testing the agent in the environments we can say that the agent E3 shows a better search behavior than any other agent, since it covers a larger area of the image and passes a maximum of 30 times over the same point in the three environments. In addition, it locate the points that cover the largest number of cells and have a distinctive search pattern, as at the start of the episode it moves down into the environment and then starts

searching. Finally, we can conclude that the values obtained in during the process do not affect in a big way in the testing results.

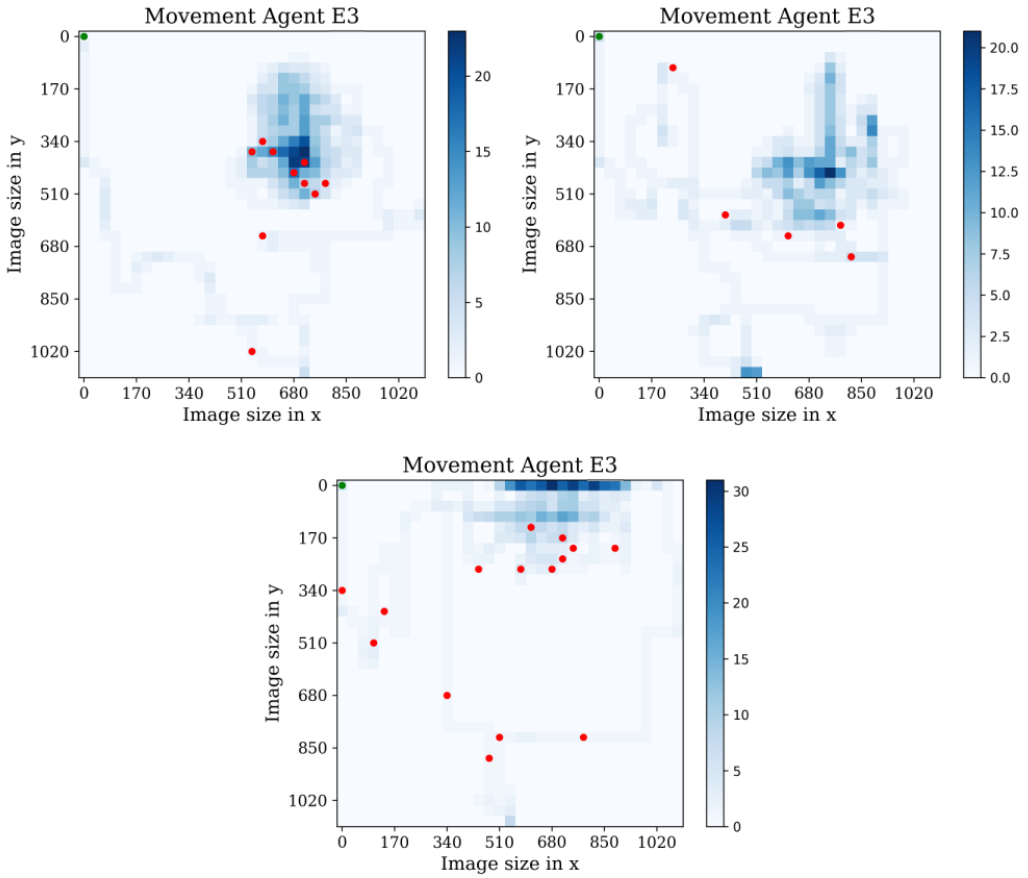



Figure 5.14: The figure shows the tracking of the agent E3, while searching in the three mentioned environments.

- Agent E4: Although agent E4 did not show high results in the mean reward values,see Fig 5.6, it can be observed that they have a good search behavior and in the three environments. The agent was able to capture a large number of points where a cell was detected. Furthermore, the found points are spread over a long area what means that the agent is exploring and exploiting the knowledge Also, when the agent gets stuck in a loop, it does not perform more than 70 steps in the same place. Finally, we can conclude that he performed a very good search in the three environments and that his search pattern is to go down in the environment and then perform the search, see Fig 5.15.

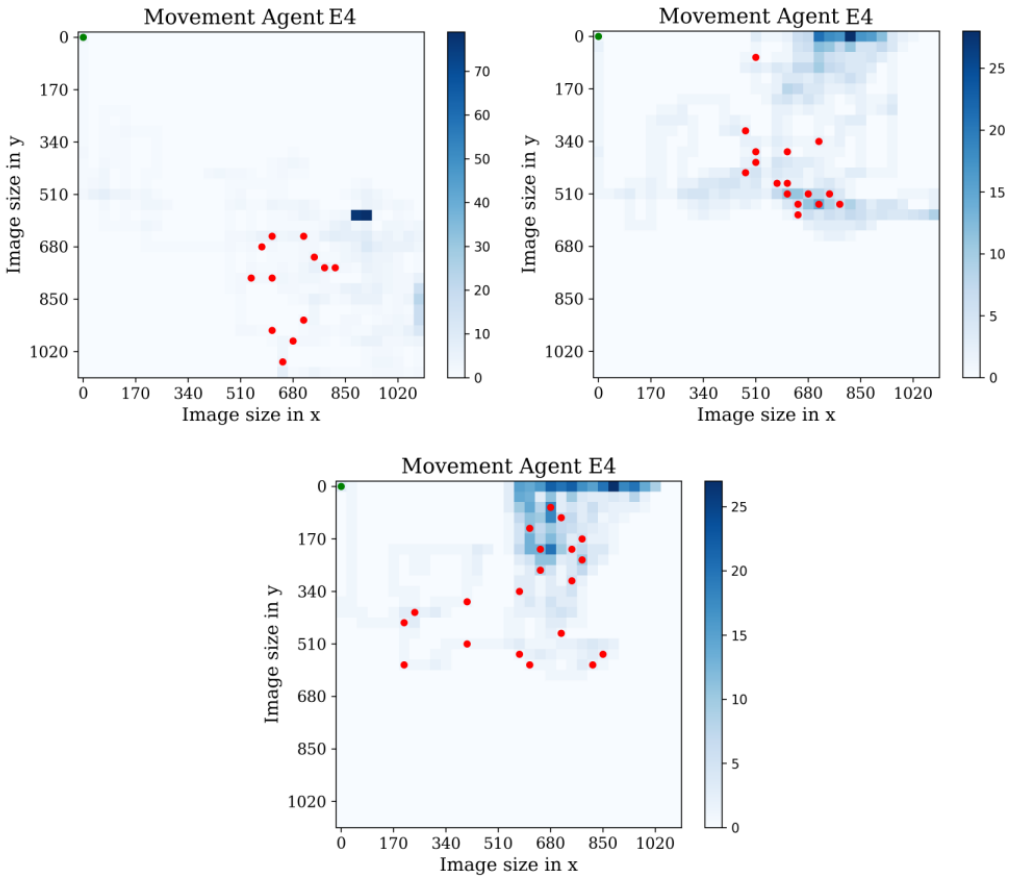



Figure 5.15: The figure shows the tracking of the agent E4, while searching in the three mentioned environments.

## 5.3 Training Method

This section concerns related to the employee approach are examined and discussed.

### 5.3.1 Sample Efficiency

Reinforcement learning is an area in which a high number of samples are often required to train an effective agent, resulting in low sample efficiency. The number of samples required to acquire good results is usually considerable and varies according to the situation and its factors. For example, an agent may require hundreds of thousands to millions of time steps to learn different control tasks [49] or in the case of Atari games which needs millions of time steps [24, 41]. The environment implemented for this project is large, images of $1080 \times 1080$ pixels were used, due to avoid loss of information which is important when the detected cell will be classified. The duration of the training process is impacted by the

size of the images, and our image size was substantial in comparison to the image sizes utilized in other reinforcement learning investigations [28, 29, 30, 41, 50]. It is crucial to ensure that the environment you choose is suitable for your particular requirements. If your objective is to create an agent with an advanced search behavior, it is beneficial to select a less complex environment with faster sampling times and smaller image sizes, such as the environments mentioned earlier [28, 29, 30, 41, 50].

### 5.3.2 Problem Difficulty

In this environment, the main challenge is that each pap smear test is considered an individual environment. These tests typically contain between 5 to 30 cells, which results in a large number of targets for the agent to locate, and their placement is randomized. So, during 1024 time steps, the agent has to locate the most number of cells before the environment resets, and a new image is introduced as the new environment. Due to constraints on training time and the fact that the best training configurations were tested, the resulting performance may not be optimal. Nevertheless, the agents exhibit a learning behavior.

### 5.3.3 Neuronal Network

Using the appropriate neural network for training is an important factor. In this project, a neural network called natural CNN was implemented, which has been proven to yield good results in training a variety of Atari games [41]. Implementing a neural network with a more sophisticated architecture could potentially enhance the agent's accuracy in locating and identifying cells. However, it is important to note that using a more advanced neural network could have a negative impact on the training process by increasing the training time. For instance, the natural CNN has an inference time of 4ms, and it takes approximately 24 hours for the agent to complete 1.2 million time steps. If the inference time is doubled, it would take the agent approximately 38 hours to achieve the same number of time steps. Doubling the inference time does not double the training time because sampling the environment still takes the same amount of time. In this project, an additional neuronal network was implemented in order to detect cells and define the

reward signal. This CNN is activated after 10 time steps and has an interference over 50ms and in some cases 100ms causing that it takes around 24 hours for performing 1 million time steps.

### 5.3.4 Performance Metric and Reward Signal

In order to define a good reward signal we use a discrete reward function which has three different types of rewards or penalties. The first penalty is applied when the agent has not any cell in the ROI, it cause that he receive a small penalty which encourage him to search for a cell. The second penalty is applied when the agent detect multiple times the same cell which means that hi is stuck in one cell. This behavior is controlled by the second penalty which give a considerable penalty when this occur. Finally, the agent receive a reward which increase depending on how many cells the agent has detected. In the experiments were defined four distinct reward function configurations that help us to determine which is the best for training. The experiments showed that high penalties could destroy the policy and build useless agents (D, H agents). In addition, it is challenging to establish a reward signal that explicitly encourages the desired behavior. This raises doubts about whether the reward is a suitable performance metric.

### 5.3.5 Reliability and Validity

In this project, it is used an algorithm [24] which implement a stochastic policy , which utilizes a probability distribution to determine the agent's actions. This inherent randomness implies that two agents trained under identical hyperparameters and conditions may exhibit different behaviors, see Fig 5.6. Therefore, to accurately assess an agent's efficiency or performance, it is necessary to train the same agent multiple times. Repeating the same experiment several times enables the creation of a performance metric with a confidence interval, which provides a more precise measure of the agent's performance [51].

Furthermore, even the n_steps hyperparameter were modified in this project, the original paper shows that PPO is a robust method [24], as proved in [51], hyperparameters can still have an influence on both training time and performance. As a result, it would have been beneficial to do hyperparameter tuning to achieve the best possible results. In addi-

tion, reinforcement learning needs a lot of samples. For instance, the PPO algorithm paper shows that for playing Atari games it was required to train the agent for 40 million time steps [24]. This is a considerable difference when compared to our agents' performance. It is obvious that our agents needs more train for achieving better results. However, time constraints do not allowed to carry out that task.

### 5.3.6 Source Criticism

This thesis primarily uses academic books and papers as sources. Some of the papers cited are only available as preprints, indicating that they have not undergone peer review. The reliability and validity of a paper may not be compromised by the lack of peer review. However, some of the preprint papers utilized in the graduate project are from the OpenAI team, who often publish their work through preprints or blog posts. Although peer review has not been conducted on these papers, they are still regarded as credible because OpenAI is a prominent and well-respected organization in the field with a remarkable track record.

Regarding the books [2, 13, 15] are renowned in their particular ares, while [12] is considered reliable due to its usage as course in many institutions. Despite being quite dated, the fact that only fundamental knowledge was derived from it means its age was not considered an issue.

## 5.4 Future Works

As mentioned in section 5.2.2, it would be interesting to resize the images to get a faster and less complex environment. Even, the image could be prepossess with some segmentation techniques such as the case of other papers where they apply threshold in the images for detecting a cell in multi-cells tests [52]. Taking into account this techniques, the agent could learn faster what is the expected behavior that may be developed. However, we need to consider if this change would impact the behavior of the agent. Additionally, if an agent is trained in a low graphical quality environment, can it perform well in a high graphical quality environment

The algorithm PPO is a very robust and common algorithm used in a vast majority of environments, but the CNN that has implemented is simple [41]. Because of our envi-

ronment needs to detect multiple objects, it could be important to use a more complex architecture in order to permit the agent to precisely identify and concentrate on targets.

Additionally, some papers use auxiliary tasks such as [42], where an agent learns how to play the video game Doom. The network used is split into two parts, with one part learning the optimal policy and the other predicting whether there is an enemy on the screen. By training the prediction network to identify enemies, the convolutional layers learn to extract features useful for detecting them. This approach could be applied to improve the agent's performance in this graduation project problem setup by having the prediction network predict the presence of a cell in the ROI.

Another approach that may help the agent in extracting relevant features is to implement transfer learning, which makes reference to transferring learned features from pre-trained network to another one. This approach is commonly used in image classification and could be useful for the current task. For instance, [50] apply this transfer learning method to train an agent who navigates through an indoor environment using images and implementing a pre-trained ResNet50 architecture [53] to extract repeating features in images.

# Chapter 6

# Conclusions

This graduation project studied the application of deep reinforcement learning for the analysis of pap smear test, where the detection of the cells is done using a ROI to capture the cells. The first question that arises is how well the agent, using a simple neural network, solved the given task. Mentioning the best agents of the 8 agents that were trained, we can comment that agent C and F demonstrated a learning of search behavior and also a fluid movement without looping actions. The performance of these agents was not perfect, and it is considered to increase the training time as suggested by most of the papers that use reinforcement learning to solve different tasks. Thus, we can also indicate that using the proposed method can solve the task.

The second question that was answered in the develop of this project was how can affect the reward function for training the agents. Even the agents D and H do not perform any action and do not shows any learning behavior, we can conclude that the reward function is one of the most sensitive parts of the agent. If the reward signal is defined incorrectly the rewards and penalties the agent will never learn what is the task that has to solve in the environment. In addition, reward signals with more than one reward function seems to be better than reward functions that stimulates the agent only in a specific action, causing that the convergence of the policy were slow. Furthermore, the use of a CNN for detecting cells were important to define the reward function which is mainly linked with what the ROI see and the CNN yield.

Furthermore, from the third stage we can mention that after more time steps the agents become more intelligent and able to solve the task proposed in the environment.

Additionally, It was shown that even if the same seed was used to train four more agents, the result will vary since reinforcement learning branch is guided by probabilities and stochastics algorithms, which causes said variations.

Finally, the artificial intelligence systems have the potential to provide numerous advantages, including enhanced effectiveness, improved safety, and reduced workload. Although the agents developed in this thesis are not yet fully functional as intelligent cells detectors, the thesis lays a foundation for future research in this field.

# Bibliography


[1] J. E. Ormrod, *Human Learning, Global Edition*, 7th ed. London, England: Pearson Education, Aug. 2015.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[3] P. A. Cohen, A. Jhingran, A. Oaknin, and L. Denny, "Cervical cancer," *The Lancet*, vol. 393, no. 10167, pp. 169–182, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014067361832470X

[4] "Cervical cancer," 2022. [Online]. Available: https://www.who.int/health-topics/cervical-cancer#tab=tab_1

[5] "What should i know about cervical cancer screening?" Dec 2021. [Online]. Available: https://www.cdc.gov/cancer/cervical/basic_info/screening.htm

[6] E. J. Liebermann, N. VanDevanter, M. J. Hammer, and M. R. Fu, "Social and cultural barriers to women's participation in pap smear screening programs in low- and middle-income latin american and caribbean countries: An integrative review," *Journal of Transcultural Nursing*, vol. 29, no. 6, pp. 591–602, Jan. 2018. [Online]. Available: https://doi.org/10.1177/1043659618755424

[7] B. Nuche-Berenguer and D. Sakellariou, "Socioeconomic determinants of cancer screening utilisation in latin america: A systematic review," *PLOS ONE*, vol. 14, no. 11, p. e0225667, Nov. 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0225667

[8] T. A. Kessler, "Cervical cancer: Prevention and early detection," *Seminars in Oncology Nursing*, vol. 33, no. 2, pp. 172–183, May 2017. [Online]. Available: https://doi.org/10.1016/j.soncn.2017.02.005

[9] "Cervical cancer ecuador 2021 country profile," Nov 2021. [Online]. Available: https://www.who.int/publications/m/item/cervical-cancer-ecu-country-profile-2021

[10] "Instituto nacional de estadística y censos," 2018. [Online]. Available: https://www.ecuadorencifras.gob.ec/institucional/home/

[11] K. Strasser-Weippl, Y. Chavarri-Guerra, C. Villarreal-Garza, B. L. Bychkovsky, M. Debiasi, P. E. R. Liedke, E. S.-P. de Celis, D. Dizon, E. Cazap, G. de Lima Lopes, D. Touya, J. S. Nunes, J. S. Louis, C. Vail, A. Bukowski, P. Ramos-Elias, K. Unger-Saldaña, D. F. Brandao, M. E. Ferreyra, S. Luciani, A. Nogueira-Rodrigues, A. F. de Carvalho Calabrich, M. G. D. Carmen, J. A. Rauh-Hain, K. Schmeler, R. Sala, and P. E. Goss, "Progress and remaining challenges for cancer control in latin america and the caribbean," *The Lancet Oncology*, vol. 16, no. 14, pp. 1405–1438, Oct. 2015. [Online]. Available: https://doi.org/10.1016/s1470-2045(15)00218-1

[12] S. Marsland, *Machine Learning - An Algorithmic Perspective.*, ser. Chapman and Hall / CRC machine learning and pattern recognition series. CRC Press, 2009.

[13] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. [Online]. Available: http://aima.cs.berkeley.edu/

[14] F. Chollet, *Deep Learning with Python*, 1st ed. USA: Manning Publications Co., 2017.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[16] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.08458

[17] J. Wu, "Introduction to convolutional neuronal networks," in *National Key Lab for Novel Software Technology. Nanjing University. China*, vol. 5, 2017.

[18] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation," *Journal of Big Data*, vol. 6, 11 2019.

[19] R. Riad, O. Teboul, D. Grangier, and N. Zeghidour, "Learning strides in convolutional neural networks," *CoRR*, vol. abs/2202.01653, 2022. [Online]. Available: https://arxiv.org/abs/2202.01653

[20] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning pooling for convolutional neural network," *Neurocomputing*, vol. 224, pp. 96–104, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216312905

[21] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *CoRR*, vol. abs/1512.00242, 2015. [Online]. Available: http://arxiv.org/abs/1512.00242

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[23] S. Khan, H. Rahmani, and S. A. A. Shah, *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan and Claypool Publishers, 2018.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[25] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," 2015.

[26] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[27] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[28] F. Moreno-Vera, "Performing deep recurrent double q-learning for atari games," 2019. [Online]. Available: https://arxiv.org/abs/1908.06040

[29] N. Desai and A. Banerjee, "Deep reinforcement learning to play space invaders," 08 2017.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[31] M. T. Rezende, R. Silva, F. de O. Bernardo, A. H. G. Tobias, P. H. C. Oliveira, T. M. Machado, C. S. Costa, F. N. S. Medeiros, D. M. Ushizima, C. M. Carneiro, and A. G. C. Bianchi, "Cric searchable image database as a public platform for conventional pap smear cytology data," *Scientific Data*, vol. 8, no. 1, Jun. 2021. [Online]. Available: https://doi.org/10.1038/s41597-021-00933-8

[32] M. A. Mohammed, F. Abdurahman, and Y. A. Ayalew, "Single-cell conventional pap smear image classification using pre-trained deep neural network architectures," *BMC Biomedical Engineering*, vol. 3, no. 1, Jun. 2021. [Online]. Available: https://doi.org/10.1186/s42490-021-00056-6

[33] B. Uzkent, C. Yeh, and S. Ermon, "Efficient object detection in large images using deep reinforcement learning," *CoRR*, vol. abs/1912.03966, 2019. [Online]. Available: http://arxiv.org/abs/1912.03966

[34] M. Alsalatie, H. Alquran, W. A. Mustafa, Y. Mohd Yacob, and A. Ali Alayed, "Analysis of cytology pap smear images based on ensemble deep learning approach," *Diagnostics*, vol. 12, no. 11, p. 2756, Nov 2022. [Online]. Available: http://dx.doi.org/10.3390/diagnostics12112756

[35] Y. Pang, B. Smola, R. T. Pu, and C. W. Michael, "Restoring satisfactory status in ThinPrep pap test specimens with too few squamous cells and containing microscopic red blood cells," *Diagnostic Cytopathology*, vol. 36, no. 10, pp. 696–700, Oct. 2008. [Online]. Available: https://doi.org/10.1002/dc.20890

[36] E. Hussain, L. B. Mahanta, H. Borah, and C. R. Das, "Liquid based-cytology pap smear dataset for automated multi-class diagnosis of pre-cancerous and cervical

cancer lesions," *Data in Brief*, vol. 30, p. 105589, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340920304832

[37] B. O. Toapanta, "Automatic high-resolution ananlysis of pap test cells," May 2021. [Online]. Available: http://repositorio.yachaytech.edu.ec/handle/123456789/336

[38] s. dhawan, "Splitting data for machine learning models," Aug 2020. [Online]. Available: https://www.geeksforgeeks.org/splitting-data-for-machine-learning-models/

[39] OpenAI, :, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1912.06680

[40] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," 2019.

[41] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[42] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," 2016. [Online]. Available: https://arxiv.org/abs/1609.05521

[43] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[44] M. The, "Generate reward function," 2022. [Online]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/define-reward-signals.html

[45] K. J. Millman and M. Aivazis, "Python for scientists and engineers," *Computing in Science Engineering*, vol. 13, no. 2, pp. 9–12, Mar. 2011. [Online]. Available: https://doi.org/10.1109/mcse.2011.36

[46] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[47] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[48] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[49] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2016, publisher Copyright: © ICLR 2016: San Juan, Puerto Rico. All Rights Reserved.; null ; Conference date: 02-05-2016 Through 04-05-2016.

[50] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," 2016. [Online]. Available: https://arxiv.org/abs/1609.05143

[51] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018. [Online]. Available: https://doi.org/10.1609/aaai.v32i1.11694

[52] S. Gautam, H. K. K., N. Jith, A. K. Sao, A. Bhavsar, and A. Natarajan, "Considerations for a pap smear image analysis system with cnn features," 2018. [Online]. Available: https://arxiv.org/abs/1806.09025

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.